Quality Embedded Intelligent Remanufacturing

THÈSE N^{O} 4522 (2009)

PRÉSENTÉE LE 1ER DÉCEMBRE 2009

À LA FACULTÉ SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DES OUTILS INFORMATIQUES POUR LA CONCEPTION ET LA PRODUCTION
PROGRAMME DOCTORAL EN SYSTÈMES DE PRODUCTION ET ROBOTIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Young Seok KIM

acceptée sur proposition du jury:

Prof. M.-O. Hongler, président du jury
Prof. P. Xirouchakis, Dr D. Kiritsis, directeurs de thèse
Prof. A. Bouras, rapporteur
Prof. M. Taisch, rapporteur
Dr M.-J. Yoo, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2009

To

My Lovely Wife Jiyoung & Precious Daughter Se eun…

Résumé

Ce travail de thèse aborde quatre thèmes principaux : réutilisation, qualité, multi-agents et intelligence. Récemment, les problèmes environnementaux ont entraîné le renforcement de la régulation et de la législation concernant les produits usagés. Cela a provoqué un intérêt croissant pour les activités de réutilisation. En effet, la qualité des produits usagés est variable, et peut même évoluer dynamiquement durant le processus de réutilisation. Par conséquent, chaque produit usagé doit être traité de manière individuelle en fonction de sa qualité. Fort heureusement, les nouvelles technologies sans fil telles que le RFID permettent aux systèmes de réutilisation d'identifier, de suivre et de gérer chaque produit usagé ainsi que ses composants et sous-assemblés (used Product and Disassembled Subassembly/Part, PDSP) de manière automatique. L'approche basée sur les systèmes multi-agents s'avère adéquate pour la gestion individuelle de chaque PDSP. En effet, un système de gestion centralisé pourrait difficilement appréhender un grand nombre important d'articles (PDSP) inhérent aux systèmes de réutilisation.

L'objectif de ce travail de thèse est de proposer un système de réutilisation orienté qualité (Quality embedded Remanufacturing System, QRS) composé d'un cadre multi-agents d'une part, et d'un mécanisme d'ordonnancement d'autre part. Tout d'abord, la thèse décrit les concepts fondamentaux, les outils de modélisation et le mécanisme d'ordonnacement proposés: les caractéristiques en termes de qualité du QRS ainsi que le cadre multi-agents. Ensuite, cette thèse propose des outils de modélisation des QRS qui supportent la représentation de la qualité des PDSP et des resources. Ces outils comprennent une représentation intuitive des systèmes de réutilisation (IRSR) d'une part, et des réseaux de Petri colorés dynamiques à deux niveaux (PTPN) d'autre part. Le premier élément s'adresse à l'utilisateur tandis que le second est destiné au système. Le cadre multi-agents est construit en se basant sur le modèle représenté à l'aide des outils présentés ci-dessus. Pour finir, cette thèse propose un mécanisme d'ordonnacement du QRS qui permet d'exécuter le cadre multi-agents développé durant cette recherche. Le mécanisme d'ordonnancement intègre un protocole de communication entre agents ainsi que des règles de dispatching dépendant de la qualité des PDSP et des ressources. Une approche s'appuyant sur les bases de connaissances est adoptée afin d'augmenter les performances du mécanisme d'ordonnancement, dont les connaissances sont acquises durant les simulations. Une heuristique est également développée afin de réduire la durée des simulations.

Mots-clés: *réutilisation, qualité, mécanisme d'ordonnacement, multi-agents.*

Abstract

This thesis is motivated from the four keywords: remanufacturing, quality, multi-agent and intelligence. Recent years' environmental problems caused tightening the regulations and legislations for used products. Therefore remanufacturing is getting more attention. The quality of used products is uncertain and even dynamically changes during the remanufacturing process, and each used product should be individually handled in a different way depending on its quality. Fortunately recent developing wireless technologies like radio frequency identification (RFID) may enable remanufacturing control systems to identify, track, and control each used product and disassembled subassembly/part (PDSP) automatically. The multi-agent approach can be a good solution for the individual control of each PDSP, because a centralized control system is not eligible to managing so many elements in the remanufacturing system.

The objective of this thesis is to propose a quality embedded remanufacturing system (QRS) which comprises a multi-agent framework and a scheduling mechanism. First, this thesis discusses the fundamental concepts for the proposed modeling tools and scheduling mechanism: the QRS quality characteristics and the multi-agent framework. As the second step, this thesis proposes QRS modeling tools which support the PDSP/resource quality representation and comprise: intuitive remanufacturing system representation (IRSR) and dynamic token two-level colored Petri-nets (DTPN). The former is designed from the user-side perspective and the latter is from the system-side perspective. The multi-agent framework is constructed based on the model represented with the proposed tools. Last, this thesis proposes a real-time scheduling mechanism for the QRS which enables the constructed framework to execute. The scheduling mechanism embeds a communication protocol among agents and dispatching rules formulated depending on the PDSP/resource quality. A knowledge-based approach is adopted to increase efficiency of the scheduling mechanism, where the knowledge is learned by simulations. A heuristic method is also proposed to reduce the simulation time.

Key words: *remanufacturing, quality, scheduling mechanism, multi-agent.*

Acknowledgements

In the first place, I would like to express my gratitude to my doctoral supervisor, Professor Paul Xirouchakis, for offering me the opportunity to start my doctoral studies in the challenging environment of the LICP in EPFL and for guiding me throughout the entire development of my research.

I thank Dr. Dimitris Kiritsis for his encouragement provided during whole stay in LICP and for his full dedication to the projects, PROMISE, that was very helpful for this research.

I address many thanks to Prof. Hong-Bae Jun and Dr. Jong-Ho Shin for their invaluable help, supporting my accommodation to Switzerland, getting an insight for projects, and stimulating discussions and research. I thank them for the trust they put in me and for their unconditional friendship. I also thank Oliver Ioan Avram and Sandeep Dhanik who are my office mate and have shared good moments. Without them, my life in EPFL would be ennui and not so exciting. To the whole lab LICP, thank you for the rich cultural environment. I've learnt a lot from all of them.

I thank my parents and sister, who helped me finding a way in my life and always trusted me, for their unconditional support and love. Without them, I wouldn't be here at this moment.

To my precious daughter Se Eun, who had hard time when she was born. You have made my life very happy, and I'm sure it will also last forever.

Last, but not least, I want to express my great gratitude and love to my wife Jiyoung Kang, who has always been by my side and devoted most of her times to me. She helped me refresh my complex mind and come back to my research world. I dedicate this dissertation to her.

Table of Contents

List of Tables

List of Figures

Glossary and Abbreviations

Used product Item that is collected to the remanufacturing shop to be remanufactured;

Subassembly Item that is disassembled from a used product; it can be disassembled more into the other subassemblies or parts;

Part Item that is disassembled from a used product or a subassembly; it cannot be disassembled anymore;

Workstation Set of resources which have the same functionalities;

Resource Human or machine which conducts remanufacturing operations directly, the operators for machines are not considered as resources;

Operation Unit work to be done for remanufacturing; for example, cleaning, refurbishing, disassembly, and so on;

Process Set of operations to remanufacture used products.

BOL Beginning Of Life;

DTPN Dynamic Token Petri-Net;

EDD Earliest Due Date;

EfS Exit from the System;

EOL End Of Life;

EtS Entrance to the System;

EWT Expected operation Waiting Time;

FCFS First Come First Serve;

ID IDentification;

IRSR Intuitive Remanufacturing System Representation;

LPOS Lowest POS;

LPOSD LPOS to Due date;

LPOSM LPOS to Modified due date;

LPOSR LPOS to Remaining total processing time;

LPOST LPOS to operation processing Time;

LPOSU LPOS to due date Urgency;

MDD Modified Due Date;

NEDD New EDD;

PDF Probability Distribution Function;

PDSP used Product and Disassembled Subassembly/Part;

POF Probability of Operation Failure;

POS Probability of Operation Success;

QRS Quality embedded Remanufacturing System.

RFID Radio Frequency IDentification;

RPQ Ratio of PDSP Qualities after an operation;

RPT Remaining Processing Time;

SPT Shortest Processing Time;

TWK Total Work content;

WINQ Work In Next Queue;

XCPN eXtended two-level Colored Petri-Nets;

List of Symbols

I. Introduction

This chapter presents the motivation, objective, and overall contents of this thesis based on the specific characteristics of the remanufacturing system compared to the conventional manufacturing system. Main assumptions of this thesis are listed at the end of this chapter.

Abbreviated terms used in this chapter:

EOL	End Of Life;
PDSP	used Product and Disassembled Subassembly/Part;
QRS	Quality embedded Remanufacturing System;
RFID	Radio Frequency IDentification.

1. Motivation

Reusing of used products is becoming more important as many countries are tightening environmental regulations or legislations in economic activities (Steinhilper 1998), and it can also be a good alternative from the perspective of cost reduction (Guide *et al.* 1998). Therefore the concerns about remanufacturing research have been recently highlighted for these reasons.

However the research on the remanufacturing system is quite different to that of the conventional manufacturing system because of the following distinguishing characteristics (Sakara 2006):

- Uncertain arrival intervals of used products;
- Uncertain lot sizes of used products;
- Diverse part compositions of used products;
- Uncertain qualities and operation times of used products and disassembled subassemblies/parts (PDSPs);
- Uncertain remanufacturing process depending qualities of PDSPs.

The core of such characteristics is the uncertain quality of the PDSPs. The PDSP quality dynamically changes during its remanufacturing process due to the defects that arise or are exposed by the remanufacturing operations; especially disassembly operations. The uncertain and dynamic quality of PDSPs affects not only the remanufactured product quality but also the remanufacturing system performance: processing time, success rate of operations, and so on. Hence the quality issues on the PDSPs should be carefully dealt with in the remanufacturing system. In this sense, this thesis deals with the quality information involved remanufacturing system, called 'quality embedded remanufacturing system (QRS)'. The detailed features of the QRS are described in section I.3.

Managing the uncertain PDSP quality in the QRS necessarily requires an individual control of each PDSP, and the multi-agent system can be a good solution for such a control. The agent approach has been recognized as a promising paradigm to overcome the limitations of conventional centralized systems in the abilities of the expansion, reconfiguration, maintenance without shutting down, and so on (Shen *et al.* 2006). This thesis defines PDSPs also as agents, which add the individual controllability of all elements in the QRS to those well known advantages of agent approaches. The individual control of PDSPs was not easy until recently, because there was no way to identify each and to get its state. But recent emerging product identification technologies like the radio frequency identification (RFID) can make it possible. Attaching RFID tags into each PDSP makes them easy: the PDSP identification and state gathering during its remanufacturing process.

To summarize, this research is motivated by the following four key words:

- remanufacturing: recent environmental issues attract the interest of people to remanufacturing;
- quality: quality characteristics is at the core and cannot be neglected in the remanufacturing system;
- multi-agent: multi-agent approach of defining PDSPs as agents can be a good solution for the QRS control;
- intelligent: emerging wireless and sensors technologies can realize a quality embedded multi-agent framework in an intelligent way.

2. Target remanufacturing system

Collected used products are partially disassembled into some subassemblies or parts. Each subassembly or part is examined to decide their end-of-life (EOL) treatments: recycle, remanufacturing, and disposal. The subassemblies/parts to be recycled are sent to material reprocessing plants depending on their ingredients, where they are physically/chemically dismantled, sorted, reprocessed, and distributed to plants using recycled materials. Those to be remanufactured are sent to remanufacturing factories. Those to be disposed of are discarded according to the environmental regulations. Some small-sized used products like mobile phones are sent to remanufacturing systems without any partial disassembly. The quality of such used products is also examined before being sent to remanufacturing systems, and only the remanufacture-able used products[1] are sent.

This research considers the processes after used products or partially disassembled subassemblies are passed into a remanufacturing system. Therefore the used products in the remanufacturing system sometimes correspond to the subassemblies which are partially disassembled from a used product and passed to the remanufacturing system.

The used products in the remanufacturing system undergo roughly three steps: disassembly, part refurbishment, and reassembly. Some parts can have additional steps like paintings, repairing, testing, and so on. Although some people (Guide *et al.* 1997 and 1998) consider only the intermediate steps between disassembly and reassembly as their remanufacturing processes, this research encloses all steps into the used product remanufacturing process in the QRS (refer to figure I.1).

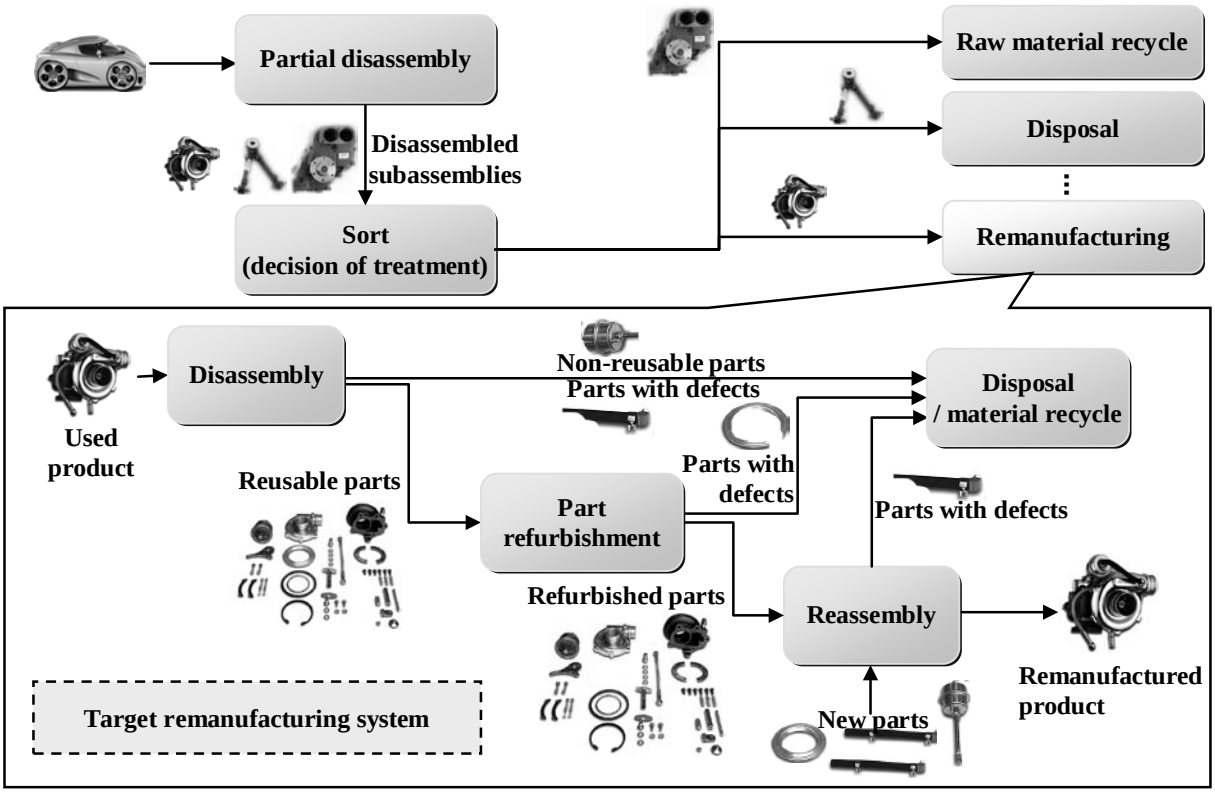


Figure I.1. Target remanufacturing system.

[1] The remanufacture-able used product means that the expected value increase of the used product by remanufacturing is bigger than the expected required remanufacturing cost.

The job-shop layout has been most widely used in the remanufacturing industry (Noble and Lim 2002). The remanufacturing shop usually handles multiple kinds of used products and the remanufacturing process of each used product is different from used product to used product depending on its quality state. Hence the remanufacturing shop should be flexible to accept various remanufacturing processes, and the job-shop style layout is more eligible than flow-shop layout. Used products in the job-shop layout remanufacturing system can visit workstations with any sequence depending on their required processes. Even though some research consider a flow-shop layout remanufacturing system like a mobile phone remanufacturing system (Franke *et al.* 2006), the remanufacturing line permits PDSPs to enter and exit at the middle of the line. Therefore it also has a characteristic of the job-shop. This research targets the job-shop facility layout to permit various remanufacturing processes.

A remanufacturing shop has one or more workstations which are composed of one or more resources. This thesis considers the workstation as group of the several resources, and the resources belongs to a workstation have the same functionalities; for example, all the resources in a disassembly workstation do disassembly operations. Even the same functionality resources can be divided into several workstations by their characteristics or physical efficiency; for instance, there can be two disassembly workstations like a workstation for the big part disassembly and a workstation for the small part disassembly. Some resources for refurbishing and cleaning are batch resources, which can simultaneously process several similar operations as a batch manner. The remanufacturing shop can handle one or more types of used products at the same time.

3. Quality embedded remanufacturing system (QRS)

This research focuses on the quality issues in the remanufacturing system. The quality of used products entered the remanufacturing system is uncertain; for example, some parts of a used product can be out of order or broken, while a used product has no defects. What makes it worse is that the exact quality measuring is impossible before disassembly because of the examination impossibility for some hidden part. The quality can even be changed by unexpected operational errors during each operation; for instance, unscrewing a bolt with a high force application during disassembly can cause a break of joined parts. It means that quality information is uncertain and changes dynamically during the remanufacturing process.

The resources on the remanufacturing shop also affect the dynamic PDSP quality. The output PDSP quality after an operation can be different depending on resources do the operation. A resource can do a certain operation very well but may do other operations badly, while another resource can do that operation not so well but other operations very well; for example, lathes are better at the refurbishment of grooves on a cylinder than robots, and an experienced person in a car engine disassembly is better at engine disassembly than beginners of the mechanical products handling who may result in longer disassembly time and higher failure frequency. Some operations like mobile phone disassembly which can be done by humans and machines also show difference depending on the resource in charge; the mobile phone disassembly by humans shows usually shorter processing time than by the machines, but the disassembly cost by machines could be cheaper than that by humans (Franke *et al.* 2006). Therefore this thesis also considers the resource performance to the dynamic PDSPs quality for the better remanufacturing system performance.

The performance can be the processing time, the mean tardiness, the percentage of operation failure, and so on. This thesis tries to integrate all related performance factors and defines the term performance as follows:

system performance	whole spent cost to remanufacture a collected used product; the remanufacturing cost consists of the operation processing cost, part disposal cost, and delay penalty cost (refer to sections VIII.2);
resource performance	processing time and failure rate for an operation (refer to section IV.1.2).

Although used products can be remanufactured with a predefined fixed process without consideration of their uncertain and dynamic characteristics, the different PDSP handling depending on its quality and the resource performance is required for the better system performance. Hence the prerequisite for the better system performance is the successive decision making with quality consideration as follows:

- which remanufacturing process should be taken;
- which operations can be skipped without effect on the remanufactured product quality;
- which resources are good for operations in the selected process;
- which PDSPs should be processed first by a resource.

Decision making for the above listed issues needs the information about the dynamic PDSP quality. Hence the PDSP quality should be examined after every operation. Such frequent quality examinations can disturb the remanufacturing progress for the mass remanufacturing of low valued used products. However the benefits may defeat the disturbances for high valued used products like many automotive subassemblies; for example, engines, transmissions, clutches, and so on.

This research defines the quality embedded remanufacturing system (QRS) as the remanufacturing system which has the following characteristics:

- high valued used products are handled;
- qualities of all PDSPs are examined after every operation;
- remanufacturing process dynamically changes depending on the real-time PDSP quality;
- resources for a PDSP remanufacturing are selected based on the PDSP/resource quality (refer to sections IV.1) and system states;
- resources select operations-to-do based on the allocated dispatching rules which dynamically changes from the perspective of the whole system performance maximization.

4. Objective and contents

This thesis proposes a quality embedded real-time scheduling mechanism based on a multi-agent remanufacturing framework. This thesis discusses the following topic first:

- QRS quality characteristics and their representation method, (chapter IV)

before discussing the proposed scheduling mechanism, because the quality is the core feature in this thesis. The scheduling mechanism is discussed as the following steps:

- propose a multi-agent framework for the QRS; (chapter V)
- propose QRS modeling tools for the multi-agent structure and agent states construction, and validate the modeling tools; (chapter VI)
- propose quality considered dispatching rules, validate the proposed rules, and find the necessity of a dynamic real-time scheduling mechanism; (chapter VII)
- propose a knowledge-based real-time scheduling mechanism, and validate the mechanism. (chapter VIII)

A QRS multi-agent framework is proposed first, because it is the fundamental concept for the discussion on the modeling and scheduling. QRS modeling tools are proposed based on the multi-agent framework. The proposed tools can represent the necessary information for a QRS control and simulation. Some quality considered dispatching rules are also proposed, because the application of new dispatching rules is the simplest way of enhancing system performance by conventional measures like flow time or tardiness. The proposed real-time scheduling mechanism contains communication protocols among agents and a knowledge-based dynamic dispatching rule allocation method. At the beginning of this thesis are review of previous related literatures (chapter II) and an example QRS introduction (chapter III) to assist the explanation efficiency.

5. Assumptions

This thesis is applicable under the following conditions:

- only family used products are remanufactured in the system; all subassemblies and parts correctly fit to the structure of remanufactured products; (Assumption I.1)
- a disposed part is instantly substituted with a corresponding new part, and parts to substitute defective parts are always available; (Assumption I.2)
- permanent joints like welding are not disassembled; (Assumption I.3)
- disassembled parts from a used product are reassembled in one operation and the reassembly is always successful; (Assumption I.4)
- quality information can be gathered without limitation; in other words, quality information is given; (Assumption I.5)
- resources have no degradation due to the operation time usage; in other words, resource performance is stable; (Assumption I.6)
- quality examination never fails; (Assumption I.7)
- statistical distribution type of the processing time of an operation is always same independent of the input/output PDSP qualities,[1] (Assumption I.8)

and the required time for the following actions is assumed to be very short in comparison to the operation

[1] The assumption is utilized to reduce the quality representation complexity in section IV.2.

processing time:

- PDSP transportation among resources and buffers; (Assumption I.9)
- disposed PDSP substitution by a new one; (Assumption I.10)
- computation in an agent and communication among agents related to the real-time scheduling mechanism; (Assumption I.11)
- PDSP quality examination right after operation processing. (Assumption I.12)

Hence the time for those actions is considered as zero. Last, the resource set up time is assumed as follows:

- the resource set up time is included in the operation time and independent with the operation handling sequence. (Assumption I.13)

II. Previous Research Review

This chapter discusses previous research related to the quality embedded remanufacturing system (QRS). Some manufacturing/remanufacturing modeling tools are explored first to find any appropriate concepts for the QRS modeling. Consecutively multi-agent frameworks and scheduling mechanisms for the manufacturing/remanufacturing system are analyzed to compare with the proposed mechanisms. Last, some miscellaneous topics are discussed to resolve sub-problems of the proposed scheduling mechanism: the batch processing scheduling and the knowledge-based approach.

Abbreviated terms used in this chapter:

IDEF	Integrated DEFinition;
PDSP	used Product and Disassembled Subassembly/Part;
QRS	Quality embedded Remanufacturing System;
UML	Unified Modeling Language;
XCPN	eXtended colored two-level Colored Petri-Nets.

1. Manufacturing/remanufacturing modeling tools

1.1. Disassembly modeling tools

To date, some research have focused on the modeling and simulation of disassembly. Many disassembly modeling tools borrowed ideas of a network representation from the research on assembly modeling. Demello and Sanderson (1990) utilized AND/OR graph, which has been widely used until now, to analyze disassembly operations. They defined all possible subassemblies/parts and assembly/disassembly relationships as nodes and arcs respectively of their AND/OR graph. They selected an optimal assembly/disassembly sequence among all possible ones by finding a path which minimizes the operation cost. Zwingmann *et al.* (2008) also utilized AND/OR graph to find optimal disassembly sequence with consideration of geometrical constraints depending on disassembly sequences.

Zussman and Zhou (1999) extended Petri-nets and developed disassembly Petri-nets to express reworks depending on success or fail of disassembly actions. They inserted a quality test place and transition after each disassembly action transition in disassembly Petri-nets for the quality dependent system control. A token in a test place diverges into different places depending on the quality test result. Thus disassembly Petri-nets enables to represent alternative disassembly sequencing depending on disassembly quality. Hsieh (2008) proposed collaborative Petri-nets to solve a cost minimization problem with consideration of resource failures.

The above modeling tools are probably suitable for the representation or analysis of an optimal disassembly sequence. But extending those tools to handle the entire remanufacturing system is not proper, since they focus only on the states of used products and disassembled subassemblies/parts (PDSPs).

1.2. Manufacturing/remanufacturing system control/simulation modeling tools

There are many modeling tools for the simulation or control of the manufacturing/remanufacturing system. Simulation systems collect mathematical information about system state transition to analyze the system performance, and the central or distributed managers in the control systems decide the next state of the systems based on their current states. In other words, the system simulation or control means the analysis and decision of system states. Therefore the manufacturing/remanufacturing system is usually regarded as an automata system, where each resource and PDSP has states such as waiting, processing, testing, and so on.

To model manufacturing/remanufacturing systems, many research utilized Petri-nets or Petri-nets extensions like timed Petri-nets, colored Petri-nets, stochastic Petri-nets, and so on. ElMekkawy and ElMaraghy (2003) used timed Petri-nets to develop a rescheduling algorithm. They modeled processing time of each operation with the time concept in transitions of timed Petri-nets. Jeng *et al.* (2002) introduced process nets with resources to represent a resource sharing manufacturing process. The process nets with resources is an extended version of resource control nets (Xie and Jeng 1999), which is also proposed to model a resource sharing manufacturing system. Reyes *et al.* (2002) modeled a flexible manufacturing system with traditional Petri-nets and applied an artificial intelligence methodology to a heuristic search algorithm for scheduling.

Some literature considered other tools like unified modeling language (UML) and integrated definition (IDEF). Marin *et al.* (2005) expressed a manufacturing process with UML, and Brandimarte *et al.* (2000) also used UML to represent a manufacturing architecture. In addition, Cho and Lee (1999) designed an information architecture for a shop floor controller with IDEF, and Ijomah and Childe (2007) modeled an overall remanufacturing process with IDEF to suggest a standardized process for an electromechanical industry.

On the other hand, some literature introduced completely new modeling tools. Xue *et al.* (2001) proposed a feature-based product representation scheme that contains production constraints depending on product structures in order to solve an optimization problem. Naso and Turchiano (2004) utilized a discrete event system, which is similar with the state model in UML, to manage a multi-agent framework in a distributed manufacturing environment. Ryan and Heavey (2006) proposed simulation activity diagram to model and simulate a manufacturing process. CIMPACT (2000) developed a simulation software, SIMAS II®, with which a manufacturing or remanufacturing system can be modeled and simulated considering stochastic processing time. The software proposed a modeling tool with traditionally fundamental modeling elements like AND/OR/XOR connectors, a workstation, a buffer, and so on. Therefore the tool can express almost all manufacturing systems.

The above mentioned tools do not address quality issues in a remanufacturing system but usually focus on the manufacturing process modeling for the processing time control. Although Mertins *et al.* (1997) tried to accommodate all information on the business process including product quality, it is not suitable to reflect the dynamic and stochastic characteristics of the QRS because of its static relationship among modeling elements.

1.3. Extended two-level colored Petri-nets (XCPN)

Although there are many modeling tools as we explored above, there are only a few research on the development of remanufacturing system modeling tools. Among the proposed remanufacturing system modeling tools, extended two-level colored Petri-nets (XCPN) (Sakara 2006) is the closest one for the system-side representation and simulation of the quality embedded remanufacturing system (QRS). The XCPN is an extended version of two-leveled timed object Petri-nets (Kis *et al.* 2000) and developed to model remanufacturing systems. The tool includes a concept of modularization which enables to model resources and PDSPs separately, hence the PDSP agent concept can be represented. The stochastic simulation of a remanufacturing system is also possible by the color and time concepts, which are inherited from colored Petri-nets and timed Petri-nets respectively. In addition, the XCPN supports the transition synchronization between resources and used products with two-level Petri-nets: a system-net and token-nets. Hence it is appropriate for consistent controls of all agents in a multi-agent system. But XCPN is also limited in terms of the dynamic control of individual elements in the QRS depending on their quality and system states; for example, it cannot represent the selection among alternative remanufacturing processes depending on the PDSP quality.

2. Multi-agent approaches for manufacturing

Rare research on the multi-agent structure application is found in the remanufacturing domain. Shen and Zhang (2008) considered a multi-agent approach, but they focused on the overall end-of-life of used products and

considered remanufacturing as just an element. Some research dealt with agent concepts for disassembly scheduling or planning, but they usually did not considered agents as active elements but just a type of information (Imamura *et al.* 2001 and Pavliska and Srovnal 2002), a module embedding a scheduling logic with collected information (Pavliska and Srovnal 2002), a module for state self-detection (Ramaswamy and Yi 1999), and so on. Although some literatures are found that discussed multi-agent based scheduling, they represented only conceptual structures without detailed mechanisms (Kopacek and Kopacek 2006 and Tateno and Kondoh 2005) or just adapted existing mechanisms in the conventional manufacturing domain like contract net (Martinez *et al.* 1997). Therefore this section summarizes multi-agent approaches in the manufacturing domain instead.

2.1. Agent-based structures

The agent approach in manufacturing has long history, and the holonic concept can be considered as the most representative systematic approach. Even though the holon concept by Koestler (1970) did not stem from the manufacturing domain, the concept starts to be popularly applied to the manufacturing domain from its introduction in the research by Hirose (1990) and his colleagues. They represented a software environment of a holonic manipulator for the product design and implementation, and some years after the prototype software was presented. The holon is usually defined as an autonomous and cooperative building block of a manufacturing system for transforming, transporting, sorting and/or validating information and physical objects (Babiceanu and Chen 2006). In the holonic environment, any information, logic, and physical object can be defined as a holon; for instance, raw material, worker, machine, product, automated guided vehicle, module in a supply chain management system, and so on. Each holon can stand alone or can be held by other holons, but all the holons cooperate to achieve a common objective. Because of the possibility of defining what cannot live by itself as a holon, the holonic approach is rather closer to the object oriented approach than the agent approach where all the agents can do something without any aid from others. For more details about the holonic concept, refer to the literature by Babiceanu and Chen (2006).

Many research show similar frameworks in the agent-based control or simulation of manufacturing systems. Valckenaers *et al.* (2007) consolidated previous research and suggested a general framework for the agent-based manufacturing, where a corresponding entity to each real-world entity is defined in a virtual-world. The defined entities emulate the real-world, and an agent system controls the entities in the virtual-world. The virtual-world is synchronized to the real-world and gives execution orders to the entities in the real-world for the manufacturing system control, while, it is modeled in off-line and simulated by a time management module for the simulation. Although the overall frameworks in the previous research are similar, their agent systems have different structures which can be classified into three types: hierarchical, heterarchical, and hybrid.

The research of Hsieh (2005), Roy *et al.* (2001), Kadar *et al.* (1998), and so on can be classified as the hierarchical structure, where a manager agent collects information from other agents, makes decision, and distributes the results. They are not so much different with the traditional centralized control, except that each resource and product information is managed by itself. Zhang *et al.* (2007) also proposed a hierarchical structure. But it can be considered rather a hybrid structure because of the agent communication and the decision making by themselves in a subsystem.

The heterarchical structures are proposed by many researchers like Macchiaroli and Riemma (2002), Krothapalli and Deshmukh (1999), Sousa and Ramos (1999), Barata *et al.* (2008), and so on. The agents in the structure communicate with each other and make their own decision by themselves without any supervisory agents. The heterarchical structure is closer to the basic idea of multi-agent systems because of the distribution of decision making roles without any centralized control.

Hybrid structures are recently suggested by some researchers like Odrey and Mejia (2003) and Wong *et al.* (2006, 2008). The action of agents in the hybrid structure is basically similar with that of the heterarchical structure; they communicate with each other and make decision. But they use the information supported by supervisor agents for their decision making. The supervisor agents manage the information about the whole system states or sometimes directly intervene in the agents' decision. Therefore the hybrid structure can be considered as the combination of the hierarchical and heterarchical structures.

More details can be found in the research by Shen *et al.* (2006).

2.2. Negotiation mechanism for manufacturing scheduling

Negotiation mechanisms applied to agent systems are tightly related with the adopted multi-agent structures. Resource or product/part agents negotiate with central manager agents which coordinate the whole system in the hierarchical structure, while resource and product/part agents negotiate with each other in the heterarchical structure. The hybrid structure maintains mediator agents which intervene in the negotiation among resource and product/part agents.

The most traditional negotiation mechanism is contract net and its modified versions which can usually be applied to the hierarchical multi-agent structure. But some of their modified versions are applied to the heterarchical multi-agent structure. Although contract net proposed by Smith (1980) was not a mechanism for the manufacturing domain, it is widely utilized for manufacturing scheduling by many people like Shen and Norrie (2001), Roy *et al.* (2001), Hsieh (2005), Cheeseman *et al.* (2005), and so on. In contract net, a production order arrival at the manufacturing system invokes a manager agent to spread required job information to all resource agents. Then the resource agents analyze the received information and bid for the jobs to the manager agent. Analyzing the collected bids from the resource agents, the manager agent selects a resource agent to do the jobs and makes contracts with the selected resource agents. Then the jobs are processed by the resources which have been contracted.

The contract net's modified versions added additional sequences or changed characteristics of contract nets. Sousa and Ramos (1999) added a re-negotiation phase to control exceptional cases like overdue, machine failure, and so on, Sandholm (2000) proposed a leveled commitment contract net concept and permitted resource agents to drop the contracts during the processing with penalty if they can get better contracts. Ouelhadj *et al.* (1999) pursued better optimal schedule with handling several orders simultaneously. Zhang *et al.* (2003) proposed more centralized negotiation protocol, where a manager agent maintains resource agent information and spreads order information only to the pre-screened resource agents.

The market-based (auction-based) negotiation mechanism is more popular in recent multi-agent manufacturing systems. The idea of market-based negotiation mechanism has originated from the idea that each agent's pursuit of its best performance can cause the semi-best performance of the whole system. Each product/part agent in a market-based negotiation mechanism spreads the job information to-be-processed and asks its required information for its decision making to all resource agents. Then resource agents analyze the information from products/parts and bids for operations. Last each product/part agent selects resources which suggested the best bids. The decision making method of each product/part agent is similar with that of the manager agent in contract net. The negotiation mechanisms of the market-based approach are usually similar except for the decision making functions and objective measures. Macchiaroli and Riemma (2002) proposed a typical market-based negotiation protocol to minimize tardiness related measures, and Kim *et al.* (1996) proposed a protocol for cost minimization. Wang *et al.* (2007) also pursued cost minimization in a cooperation environment among small to medium sized enterprises. Lim and Zhang (2004) involved the manager agent in the protocol to mediate negotiation between resource and product/part agents. The manager agent collects the resources' bids and selects resources for each product/part agent based on the priority value of the total production time divided by delivery time, hence it can reduce the repetition times of request and rejection cycles.

The market-based approaches usually lack consideration of the overall system states, and the hybrid negotiation mechanism is introduced to overcome such deficiency. Wong *et al.* (2006) proposed a hybrid negotiation mechanism originated from contract net, where a supervisory agent assist product/part agent's negotiation based on a heterarchical market-based mechanism. The supervisor agent does not intervene in normal negotiations but only in some exceptional cases; the supervisor agent continuously monitors product/part agents and intervenes only in case product/part agent' decision can make the system too unbalanced; for example, parts are concentrated to a certain resource over a certain limit. When a supervisor agent intervenes in the product/part agent's decision making, product/part agent accepts the supervisor's advice instead of pursuing its own objective. Odrey and Mejia (2003) also proposed a hybrid negotiation mechanism to overcome abnormal cases of resources like machine breakdown.

Refer to the literature by Reaidy *et al.* (2006) for more detailed information on negotiation mechanisms.

3. Remanufacturing scheduling

While there is plenty of research on manufacturing scheduling, only a limited numbers of publications dealt with remanufacturing scheduling. Most research on remanufacturing scheduling is done by Guide and his colleagues (Guide 1996 and Guide *et al.* 1997, 1998, 2000, and 2005). Although remanufacturing includes the whole process from used product disassembly to remanufactured product testing, they confined remanufacturing only to the process in-between disassembly and reassembly. At first, they adopted the drum-buffer-rope approach to improve the performance of tardiness, throughput, work in process, and sojourn time (Guide 1996), where they arranged the buffer size before the assembly operation and the part release time after disassembly operation depending on the processing time. They also analyzed the performance of some strategies classified by the two criteria: release mechanisms of disassembled parts and dispatching rules to be applied to the resources. After that, they proposed expediting dispatching rules depending on the ratio of waiting parts in the reassembly buffer

(Guide *et al.* 1997 and 1998). They also analyzed the system performance depending on the used product structure complexity and the resource utilization level to find that the releasing mechanism does not affect the performance and that the best performance dispatching rule is different depending on the used product structure complexity. They recently proposed an analytical approach based on the queuing theory to find best performance dispatching rule for a remanufacturing facility (Guide *et al.* 2005).

Some research considered repair shops as the remanufacturing system, because the processes in a repair shop are similar with that of a remanufacturing system; a product is disassembled, parts are fixed and treated, and the parts are reassembled. But the objective of the repair shops is usually minimizing product down time, therefore the overall scheduling approaches are different with what is considered in the remanufacturing system. Stanfield *et al.* (2006) proposed a heuristic method to find a semi-optimal solution of the product releasing sequence and time. Guide *et al.* (2000) also dealt with repair shop scheduling; they analyzed the flow time and tardiness performance of each dispatching rule in a repair shop.

Some research dealt with remanufacturing scheduling from an extraordinary perspective on the remanufacturing system. Voutsinas and Pappis (2002) considered the remanufacturing system as the whole supply chain from the used product collection to the re-sail to customers, where processing time increases and the value of remanufactured products deteriorates by time. They proposed a heuristic algorithm to find a best job sequence.

As explored above, there is some research on remanufacturing scheduling. But no research directly dealt with dynamic real-time remanufacturing scheduling, which enables the remanufacturing process of each used product and the dispatching strategy of each resource changes dynamically depending on the remanufacturing system states in real-time. Even though expediting dispatching rules (Guide *et al.* 1998) can be considered as a dynamic scheduling method from a broad perspective, they left dynamic remanufacturing scheduling as future research (Guide *et al.* 2005).

4. Miscellaneous issues

Some sub-issues should be discussed for QRS scheduling in addition to the above explored main issues. This section explores previous research on the two important sub-problems of the QRS scheduling mechanism: batch processing scheduling and knowledge-based scheduling.

4.1. Batch processing scheduling

The batch processing means that the similar or same operations of some parts are simultaneously processed by an identical resource, and batch processing scheduling deals with the strategies for the job combinations to be simultaneously processed and the dispatching sequence of the combinations. Cleaning or some refurbishment operations in the remanufacturing system are usually processed by batch resources. Hence batch resources scheduling should be embedded in the remanufacturing system scheduling mechanism.

The research domains on batch processing scheduling are concentrated in the two areas: burn-in operation

scheduling in semiconductor manufacturing and package allocation strategies in logistics. Research for the other domains are seldom found. Although Li *et al.* (2007) handled batch processing in remanufacturing, they focused on a lot sizing problem in an integrated system considering manufacturing, remanufacturing, and emergency procurement/outsourcing. Therefore this section mainly discusses the research on those domains.

Ikura and Gimple (1986) proposed an algorithm to minimize the completion time of a single batch resource under given releasing time, and proved that their algorithm found an optimal solution. While they considered only one product type, many other research handled problems of multiple product types with dynamic programming (Lee *et al.* 1992, Chandru *et al.* 1993, and Sung *et al.* 2002) or other mathematical methods (Poon and Zhang 2004). Although an optimal solution can be found for the problem of given job arrival and processing time, the complexity of the optimal solution finding logic is too high. Consequently they usually applied heuristics to find semi-optimal solutions and compared with the optimal solutions for small size problems.

While the above research considered given job arrival time, some research considered unknown job arrival time. Van der Zee (2004) solved the batch processing problem in a real-time situation. He gave a binary decision to a machine: either the machine should start a batch process now or it should wait more. He proposed a look-ahead strategy to estimate next job arrivals. Glassey and Weng (1991) and Fowler *et al.* (1992) developed heuristic methods with the assumption that future job arrival information could be gathered from other systems.

Some research handled the problem from the perspective of pure mathematics and proposed mathematical solutions for a queuing system of dynamic arrivals and bulk services (Neuts 1967, Barnett and Kleitman 1978, and Deb and Serfozo 1973). They assumed job arrival intervals as a statistical distribution. Hence their work can be adopted in case job arrival time for a batch resource in the remanufacturing system shows a statistical form.

As explored above, previous research usually did not deal with an overall manufacturing line but the batch processing itself with a localized view point, because the burn-in operation is usually at the end of whole processing in semiconductor domain, and package allocation itself is the whole problem in logistics domain. Nevertheless some research dealt with whole systems containing batch resources. Neale and Duenyas (2000) developed a heuristic approach to schedule a manufacturing system comprising a batch resource and non batch machines, and Van der Zee (2002) developed a dynamic scheduling method for a flow shop with a batch resource with his look-ahead strategy.

Refer to the survey report by Mathirajan and Sivakumar (2006) for more detailed information.

4.2. Knowledge-based manufacturing scheduling

Applying proper manufacturing/remanufacturing schedules depending on the real-time system state can make the system more effective. The knowledge-based scheduling means utilization of pre-acquired knowledge when a system decides the proper schedule for the performance maximization. The knowledge is usually accumulated by trainings before or during the system execution,

There are very few publications on knowledge-based remanufacturing. Even the literatures on knowledge-based approaches for the remanufacturing system did not dealt with remanufacturing scheduling but remanufacturing

planning (Song *et al.* 2005) or disassembly sequencing (Veerakamolmal and Gupta 2002). Therefore this section explores research on knowledge-based manufacturing scheduling.

The main issues of knowledge-based approaches can be divided into the following three: the type of knowledge, the knowledge representation method, and the way of knowledge generation/accumulation. With regard to the three issues, knowledge-based approaches are usually applied with the following steps; the knowledge to be utilized for a system is defined, a knowledge representation method is selected, the knowledge is generated or accumulated by training, and the built up knowledge is utilized in the manufacturing scheduling. Ideas about overall knowledge-based approaches for manufacturing scheduling could be found from the research by McPherson and White (2006) and Dawood (1996).

In the manufacturing scheduling domain, knowledge is popularly considered as the best dispatching rules at a system state (El-Bouri and Shah 2006, Liu and Dong 1996, Trappey *et al.* 2007, and Yildirim *et al.* 2006). The research with other perspectives also found, where the knowledge is defined as resource allocations to waiting jobs (Yildirim *el al.* 2006), resource operation histories (Shen *et al.* 2007), whole manufacturing schedules enclosing process plans and a resource reselection policy for busy jobs (Zhang and Chen 1999), and so on.

The above mentioned knowledge can be represented in many ways, and the most popular representation methods are condensed into the following two: the artificial neural network approach and the rule-based approach.

Artificial neural networks perform calculations with weight factors which are allocated to each node on the network. Input parameters for the network are usually variables representing system states like the number of products/parts in the system, the resource utilization level, and so on. Therefore the artificial neural network returns a solution derived based on the embedded calculation logic with the learned weights depending on gathered system state variables. The weights are set by training with the samples representing the relationship among system states and their corresponding solutions. The corresponding solution for each system state is usually found by simulation approaches. Some advanced artificial neural network approaches permit the update of their weights by learning from the real-time execution results. Manufacturing scheduling with artificial neural network approach can be found in the research by Haq and Ramanan (2006), El-Bouri and Shah (2006), Yildirim *el al.* (2006), Liu and Dong (1996), and so on.

The knowledge in rule-based approaches keeps a decision logic depending on manufacturing system states usually in the form of the decision tree. A solution is driven by consecutive multiple step comparisons of current system states with solution set filtering criteria. Passing each step reduces the solution set size and concludes one solution last. The logic can be gathered by simulation for each system state or analyzing system execution results, but sometimes it is updated by field experts' knowledge (Kerr and Ebsary 1998). Rule-based approaches for manufacturing scheduling can be found more in the research by Ip (1997), Odonoghue *et al.* (1994), Shnits and Sinreich (2006), Trappey *et al.* (2007), and so on.

Refer to the work by Akyol and Bayhan (2007) and Pflughoeft *et al.* (1996) for more detailed information.

III. QRS Example

This chapter represents a simple example of the quality embedded remanufacturing system (QRS), where two kinds of used products are remanufactured in a remanufacturing shop equipped with 5 kinds of workstations. This example QRS is to assist understanding this thesis which explains the proposed modeling tools and scheduling method with a part of this example. The numerical values like operation processing time or disposal cost of the example are arbitrarily selected based on the following information; disassembly consumes about 40% of total remanufacturing time, and remanufacturing can reduce PDSP cost by about 50-75% (Krill and Thurston 2005). The information described in this chapter is the least information for the complete application of the proposed scheduling mechanism in this thesis. The identifications (IDs) of workstations, resources, and used product and disassembled subassemblies/parts (PDSPs) are marked with '[' and ']' enclosing brackets.

Abbreviated terms used in this chapter:

EfS	Exit from System;
EtS	Entrance to System;
ID	IDentification;
IRSR	Intuitive Remanufacturing System Representation;
PDSP	used Product and Disassembled Subassembly/Part;
POF	Probability of Operation Failure;
QRS	Quality embedded Remanufacturing System.

1. Remanufacturing shop

The remanufacturing shop of the example is a job-shop which comprises 3 buffers and 8 workstations (refer to figure III.1):

- an input buffer, an output buffer, a central buffer,
- and 3 disassembly workstations, a refurbishing workstation, 2 cleaning workstations, a reassembly workstation, and a test workstation.

1.1. Buffers

Each buffer has its capacity (in number of PDSPs), which is denoted with a circled C below the box for a buffer in figure III.1. The buffers [BF1] and [BF3] are the places for arrived used products and remanufactured products respectively. The remanufactured products are transported into some outside warehouses periodically. The buffer [BF2] contains all PDSPs for remanufacturing. The used products and disassembled subassemblies/parts (PDSPs) in [BF2] can go to or come from any workstations, hence any kind of remanufacturing process is possible; in other words, PDSPs have no sequence restriction in processing their required operations. Table III.1 describes the detailed buffer specifications. The type of buffers will be explained in section VI.2.1.

1.2. Workstations

Workstations have one or more resources which have the same functionalities, but the resource performance can be different even in an identical workstation. The belonging resources' identifications (IDs) of a workstation are marked below the box with circled R in figure III.1 like the two resources [RC1_1] and [RC1_2] of the workstation [WS1].

The resources in workstations [WS1], [WS4], [WS5], and [WS6] are machines, while the resources in the workstations [WS2], [WS3], [WS7], and [WS8] are humans as marked in the left boxes in the workstation boxes with the characters 'M' and 'H' respectively in the figure. Each character means that the workstation comprises machine or human type resources, respectively. The resources in [WS4] and [WS5] are batch resources. The circled B symbol below the box indicates it. The detailed resources' batch capacities are described in section III.4. Table III.2 describes the detailed workstation specifications. The physical and functional type of workstations will be explained in section VI.2.1.

2. Remanufacturing products

The above remanufacturing shop remanufactures two kinds of used products: pistons and camboxes (refer to figures III.2 and III.4), which are defined as the used product [A] and [B] respectively. The used products and their structure are adopted from two subassemblies of the automobile Maserati (Enrico 2008).

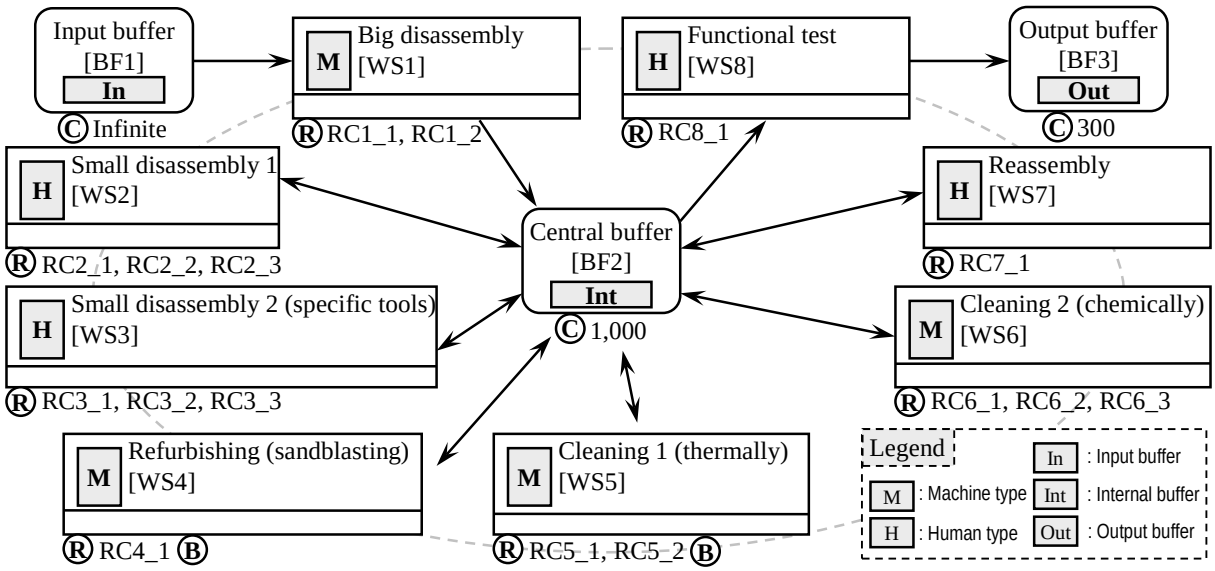


Figure III.1. Workstations and buffers composing the remanufacturing shop in the example QRS.

Table III.1. Characteristics of buffers on the remanufacturing shop in the example QRS.

ID	Name	Type[a]	Capacity	Role
BF1	Input buffer	*EtS*	Infinite[b]	collection place of input used products
BF2	Central buffer	*Internal*	1,000	waiting place of PDSPs for the next operation
BF3	Output buffer	*EfS*	300	temporal storing place of remanufactured products used as a transition to warehouse transporting

[a] *EtS*: Entrance to the System, *EfS*: Exit from the System
[b] The infinite capacity of a buffer is impossible in reality, hence the infinite capacity can be considered as a very big amount capacity.

Table III.2. Characteristics of workstations on the remanufacturing shop in the example QRS.

		Type		Resources		Related buffers		
ID	Name	Physical	Functional	ID	Batch	Input	Output	Role
WS1	Big disassembly	machine	disassembly	RC1_1 RC1_2	NO	BF1	BF2	Disassemble used products into subassemblies/parts
WS2	Small disassembly 1	human	disassembly	RC2_1 RC2_2 RC2_3	NO	BF2	BF2	Disassemble subassemblies into subassemblies/parts
WS3	Small disassembly 2 (specific tools)	human	disassembly	RC3_1 RC3_2 RC3_3	NO	BF2	BF2	Disassemble subassemblies with specially designed tools for disassembly
WS4	Refurbishing (sandblasting)	machine	refurbishing	RC4_1	YES	BF2	BF2	Refurbish parts surface by sandblasting
WS5	Cleaning 1 (thermal)	machine	cleaning	RC5_1 RC5_2	YES	BF2	BF2	Clean parts thermally
WS6	Cleaning 2 (chemical)	machine	cleaning	RC6_1 RC6_2 RC6_3	NO	BF2	BF2	Clean parts chemically like with a solvent
WS7	Reassembly	human	reassembly	RC7_1	NO	BF2	BF2	Reassemble disassembled and refurbished parts
WS8	Functional test	human	testing	RC8_1	NO	BF2	BF3	Test remanufactured products whether they meet the requirements or not

2.1. Remanufacturing product [A] - piston

The example used product [A], a piston, is composed of 5 parts: three piston rings, a piston head, and a crankshaft (refer to figure III.2(a)), IDs of which are sequentially [A1], [A2], and [A3]. The three piston rings are considered as one part for simplicity. The piston rings and head are joined by press fitting without any joining elements, while the piston head and the crankshaft are joined with a pin and two pin clips. The IDs of the two joints are sequentially [AJ1] and [AJ2] as in the liaison graph of [A] (refer to figure III.2(b)) which represents the connection relationships among composing parts.

Figure III.3 represents the remanufacturing processes of [A] where the disassembled part in the process is represented with an AND/OR graph. This thesis considers that PDSP quality is examined after an operation and sometimes the operation can be redone if the quality examination result is not satisfactory. But the quality examination and rework operations are not depicted in the figure, because this thesis considers them as internal sub-operations of the represented operations; it will be explained in chapter VI in detail. Appendix F shows all the alternative processes of [A], which is represented by the graphical notation of the intuitive remanufacturing system representation (IRSR) which will be explained in chapter VI.

Each part of [A] can have the following defects:

[A1] cracks, rust, wear;
[A2] surface scratch, cracks, weak wear of the grooves, strong wear of the grooves;
[A3] cracks.

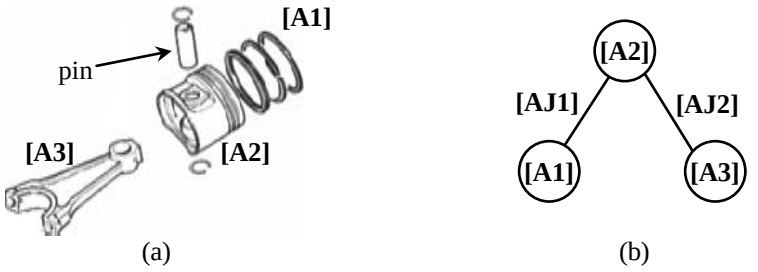


Figure III.2. Used product [A] in the example QRS, a piston subassembly: (a) three composing parts and (b) their liaison graph.

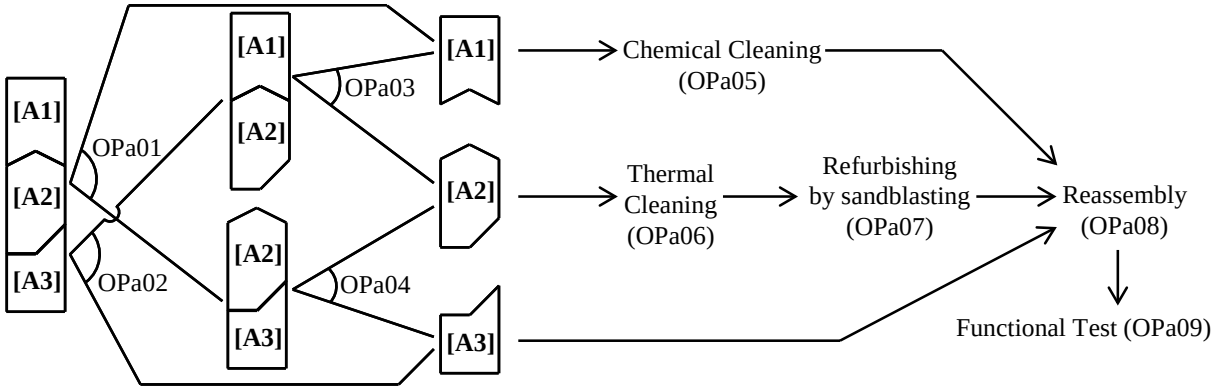


Figure III.3. Remanufacturing process of used product [A] piston in the example QRS.

Table III.3. Effects of joint defects on disassembly operations of used products in the example QRS.

		Effect to (%)[a]		
Joint ID	Possible defects	disassembly time	POF	joined parts
AJ1	Adherence of the parts by rust	60 ↑	↑↑	Necessarily cause the break of [A1]
AJ2	Adherence of the pin and pin cli	80 ↑	40 ↓	-
	Break of the pin clips	55 ↓	↓↓	-
BJ1	Abrasion of the bolt	120 ↑	370 ↑	-
BJ2	-	-	-	-
BJ3	Abrasion of the bolts	60 ↑	30 ↑	-
	Adherence of the parts	30 ↑	50 ↑	Cause the deformation of [B4] by 20%
BJ4	Tight fitting	50 ↑	15 ↑	Cause the deformation of [B4] by 20%
BJ5	Break of the bolt	80 ↑	↑↑	Necessarily cause the break of [B5]

[a] ↑, ↓: Increase or decrease of the processing time or POF of disassembly operations (for instance, 60 ↑ in the third column means the disassembly operation processing time of a defect case is 60% longer than that of no defect cases and 40 ↓ in the forth column means POF decrease by 40%), ↑↑: POF increases to 1, i.e., the disassembly operation always fails, ↓↓: POF decreases to 0, i.e., the disassembly operation always succeeds.

A rusted [A1] can be used for reassembly after removing the rust by chemical cleaning (OPa05), while [A1] having other defects should be disposed of because of the irrecoverableness of the cracked or worn [A1]. [A2] should be cleaned (OPa06) in any case. After cleaning, the surface scratch and weak wear of the grooves of [A2] can be recovered by sandblasting (OPa07). But the strong wear of the grooves or crack cannot be recovered, and [A2] having such defects should be disposed of right after the quality recognition. The cracked [A3] should be disposed of. All the disposed parts are immediately substituted with corresponding new ones without limitation (refer to assumptions I.2 and I.10). The refurbished parts [A1], [A2], and [A3] having no defects are reassembled again (OPa08).

Each joint of [A] can have the following defects:

[AJ1] adherence of the parts by rust;
[AJ2] adherence of the pin and pin clips, break of the pin clips.

The joint defects affect the processing time and probability of operation failure (POF) during disassembly. The operation failure means that the operation is uncompleted or completed with additional defects; in other words, the disassembly operation success corresponds to the case the joined parts are completely separated with the preservation of the quality of each part and joint, and all the other cases are failure cases. More details will be discussed in chapter IV (refer to figure IV.2).

The adherence of the two parts [A1] and [A2] by the rust of [AJ1] necessarily causes the break of [A1] as well as increases disassembly time. It means that the disassembly of adhered [A1] and [A2] always fails; in other words, the POF of the case is 100%. In case of the adherence of the pin and pin clips of [AJ2], the joined [A2] and [A3] can be disassembled only by breaking the pin. Therefore the disassembly time increases, while the POF is lower than the disassembly without pin breaking. The break of the pin clips makes [AJ2] cutting easier than other cases, because detaching the pin clips from the pin is not required. Consequently it decreases the processing time as well as the POF. Table III.3 synthesizes the detailed effects depending on each joint defect with numerical values.

Table III.4. Operation characteristics of used products in the example QRS.

Operation ID	Workstations in charge	Processing time distribution of non-defect input PDSP (minute)[a]	POF (%)	Limit number of rework times
OPa01	WS2	Normal(9.6, 1.0)	6.2	-
OPa02	WS2	Normal(9.3, 0.9)	9.8	-
OPa03	WS2	Normal(6.4, 0.6)	8.6	-
OPa04	WS2, WS3	Normal(8.9, 0.9)	11.0	-
OPa05	WS6	Deterministic(6.4)	5.0	2
OPa06	WS5	Deterministic(16.2)	3.2	2
OPa07	WS4	Normal(4.4, 0.4)	15.7	1
OPa08	WS7	Normal(8.7, 0.9)	0.0	0
OPa09	WS8	Deterministic(2.8)	0.0	0
OPb01	WS1	Normal(2.3, 0.2)	6.2	-
OPb02	WS1	Normal(12.6, 1.3)	27.1	-
OPb03	WS2, WS3	Normal(6.1, 0.6)	10.3	-
OPb04	WS1	Normal(10.3, 1.0)	29.8	-
OPb05	WS2, WS3	Normal(6.9, 0.7)	12.2	-
OPb06	WS2	Normal(1.9, 0.2)	3.4	-
OPb07	WS5	Deterministic(30.4)	10.5	2
OPb08	WS6	Deterministic(20.8)	19.7	1
OPb09	WS4	Normal(9.7, 1.0)	13.0	1
OPb10	WS4	Normal(8.6, 0.9)	10.5	2
OPb11	WS6	Deterministic(18.9)	10.0	1
OPb12	WS5	Deterministic(13.7)	0.0	2
OPb13	WS5	Deterministic(18.8)	0.0	2
OPb14	WS7	Normal(11.5, 1.2)	0.0	0
OPb15	WS8	Deterministic(3.1)	0.0	0

[a] Normal(α, β): Normal distribution with the average and standard deviation of α and β, deterministic(α): Deterministic distribution with the average of α.
[*] The PDFs and POFs refer to the operations and represent a reference level of performance for the corresponding resources; exact resource performance is detailed in table III.6.

Table III.4 represents the operation characteristics of [A]; workstations in charge, average processing time, POF, and the limit number of rework times. The processing time distribution and POF in the table are for the representative case: operation success with no defect input PDSPs (refer to figure IV.2). Appendix E will represent the detailed values for other cases with the proposed modeling tool in chapter VI: intuitive remanufacturing system representation (IRSR), and the usage of the values and case classification will be discussed in chapter IV. For the case an operation can be processed by two or more resources, a resource selection method will be proposed in chapter VIII.

The workstation [WS3] is equipped with specific disassembly tools as mentioned above. Those tools decrease 20% of the average processing time and 30% of the POF of the joined [A2] and [A3] disassembly (OPa02 and OPa04). Therefore the average processing time and POF of OPa04 by the resource in [WS3] is 7.1 minutes (= 8.9×(1-0.2)) and 7.7 % (= 11.0×(1-0.3)) respectively. Some operations show extraordinary processing time at some specific cases; the separation failure case of [A1] from [A] in the disassembly operation OPa01 usually shows 30% longer average processing time, because the separation failure is usually resulted after longer time separation trials. While the [A1] cracking during the operation OPa01 and the [A2] cracking during OPa07 result in 30% and 20% shorter average processing time respectively, because the operation stops immediately in case of the cracking occurrence. The usage of this information will be discussed in section IV.2.1.1.

Table III.5. Statistical information of used product arrivals in the example QRS.

Used product	Arrival interval (minute)	Lot size at arrival time (unit)	Quality at arrival time	
[A]	Poisson (500)	Normal (24.5, 2.5)	70 %	No defect
			20 %	[AJ1] and [AJ2] adherence
			10 %	[AJ2] break
[B]	Poisson (800)	Normal (22.4, 4.4)	90 %	No defect
			10 %	[BJ1] abrasion

One or more [A] simultaneously arrive at the remanufacturing shop in a bucket, called the bulk arrival in this thesis. The number of [A] at each bulk arrival follows a normal distribution, the mean and standard deviation of which are 24.5 units and 2.5 units respectively. The bulk arrivals follow a Poisson distribution with the average arrival interval of 500 minutes. 70% of the collected [A] have no defects, but 20% and 10% of them have the adherence of two joints [AJ1] and [AJ2] and the break of pin clips in [AJ2] respectively. The arrival information of [A] is summarized in table III.5.

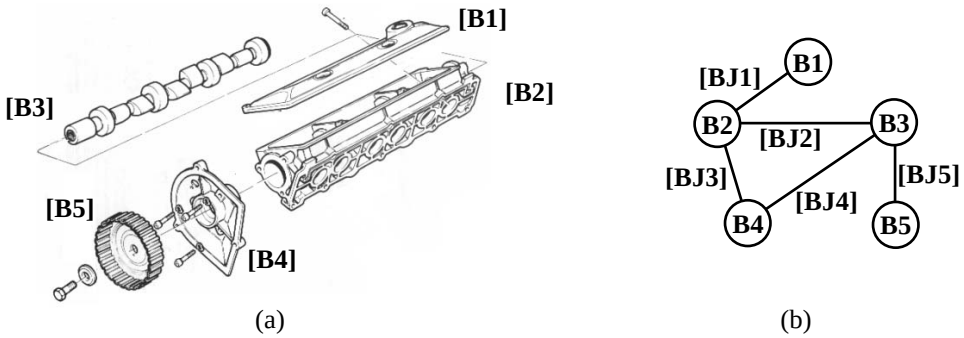
2.2. Remanufacturing product [B] - cambox

The example used product [B], a cambox, comprises 5 parts: a cambox cover, a cambox base, a camshaft, a cambox extension, and a timing pulley (refer to figure III.4(a)). Their IDs are sequentially [B1], [B2], [B3], [B4], and [B5]. The cambox base is joined with the cambox cover and the cambox extension with bolts, where the IDs of the two joints are sequentially [BJ1] and [BJ2]. The camshaft and the timing pulley are also joined by a bolt, but each end of the camshaft is loosely aligned with the cambox base and extension. The corresponding IDs of the three joints are sequentially [JB3], [JB4], and [JB5]. The liaison graph in figure III.4(b) represents the relationship among parts and their joints. Figure III.5 represents the remanufacturing processes of [B] in the same way with that of [A].

Each part of [B] can have the following defects:

[B1] cracks, rust;
[B2] wear of the hole, cracks;
[B3] cracks, wear of the both ends, deformation;
[B4] deformation, rust;
[B5] cracks, abrasion of the teeth, rust.

Any cracked parts of [B] should be directly disposed of. The thermal cleaning (OPb07) is required only for the rusted [B1]. The non cracked [B2] is chemically cleaned (OPb08) and its hole is examined. The [B2] with the hole worn too much is disposed of, and the other [B2] is refurbished by sandblasting (OPb09). The part [B3] is measured right after disassembly (OPb06), and it is disposed of in the following two cases: both its ends are worn too much or it is deformed. Otherwise [B3] is refurbished (OPb10) and chemically cleaned (OPb11). The parts [B4] and [B5] are always thermally cleaned (OPb12 and OPb13 respectively) independent of the rust on their surface except for the case of the deformation of [B4] and the teeth abrasion of [B5]. [B4] and [B5] are directly disposed of in such exceptional cases. All the disposed parts are immediately substituted with corresponding new one without limitation like the parts of [A].



(a) (b)

Figure III.4. Used product [B] in the example QRS, cambox subassembly: (a) five composing parts and (b) their liaison graph.

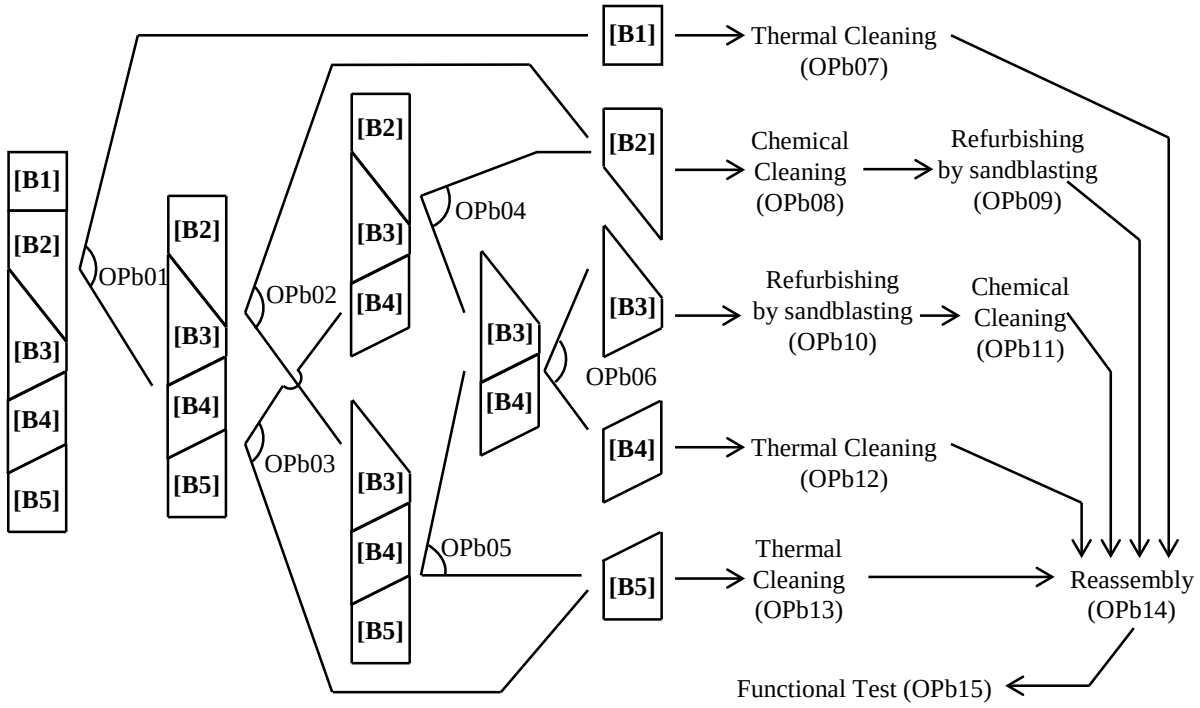


Figure III.5. Remanufacturing process of product [B] cambox in the example QRS.

Each joint of [B] can have the following defects:

[BJ1] abrasion of the bolt;
[BJ3] abrasion of the bolts, adherence of the parts;
[BJ4] tight fitting;
[BJ5] break of the bolt.

The joint [BJ2] never shows any defects. All the above defects increase the disassembly processing time and POF. The detailed information of the effects to the disassembly operations are listed in table III.3.

The operation characteristics of [B] remanufacturing are in the table III.4. The effects of using specific tools in [WS3] for the operations OPb03 and OPb05 is 40% and 20% decrease of the processing time and POF respectively. The arrival information of [B] is also found in table III.5.

3. Resource performance difference

This thesis considers the performance difference among resources in processing operations which the next chapter IV will discuss in detail. Table III.6 shows the resource performance difference for the operations in the example quality embedded remanufacturing system (QRS). The processing time performance difference indicates how much longer a resource processes an operation compared to the average processing time of the operation reported in table III.4. Hence the processing time of a resource for an operation can be calculated by multiplying the average processing time of an operation and the resource performance difference in table III.6; for example, the processing time of the operation OPa01 by the resource RC2_1 is 11.81 (= 9.6×1.23) minutes. In a similar way, the POF performance difference means how much bigger the POF is from the POFs reported in table III.4. The table shows only the resource and operation combinations which show different performance with the average case. The performance difference in case of using specific tools in [WS3] is also reflected in the values.

4. Capacity of batch resources

Resources in the workstations [WS4] and [WS5] process operations as batch (refer to figure III.1). [WS4] can handle [A2], [B2], and [B3], while [WS5] can handle [A2], [B1], [B4], and [B5]. But each workstation cannot simultaneously process parts from different used product types; for example, [WS5] can clean [B1], [B4], and [B5] at the same time, but cannot clean [A2] together with [B1], [B4], or [B5].

Table III.6. Performance difference among resources for capable operations in the example QRS.

Operation ID	Resource ID	Processing time performance difference	POF performance difference
OPa01	RC2_1	1.23	0.80
	RC2_2	0.85	0.86
	RC2_3	1.38	1.15
OPa02	RC2_1	0.90	0.87
	RC2_2	1.31	0.72
OPa04	RC2_1	1.03	0.92
	RC2_3	0.94	1.15
	RC3_1	0.75	0.67
	RC3_2	0.82	0.73
	RC3_3	0.83	0.70
OPa06	RC5_1	1.10	-
	RC5_2	0.93	0.91
OPb01	RC1_1	1.41	1.07
	RC1_2	0.85	0.73
OPb03	RC3_1	0.62	0.74
	RC3_2	0.60	0.84
	RC3_3	0.58	0.82
OPb05	RC3_1	0.62	0.74
	RC3_2	0.59	0.83
	RC3_3	0.59	0.83
OPb06	RC2_1	-	0.82
	RC2_2	1.22	-
	RC2_3	1.18	1.21
OPb07	RC5_1	1.10	1.13
	RC5_2	0.95	0.79

Table III.7. Batch resource capacity depending on parts in the example QRS.

Resource	Capacity for each part[a]					
	A2	B1	B2	B3	B4	B5
RC4_1	40 (0.025)	-	16 (0.0625)	20 (0.05)	-	-
RC5_1	50 (0.02)	16 (0.0625)	-	-	50 (0.02)	50 (0.02)
RC5_2	25 (0.04)	10 (0.1)	-	-	25 (0.04)	25 (0.04)

[a] The value in parenthesis is the occupation portion of each part of the capacity of each resource.

Table III.8. Disposal cost of each part in the example QRS.

Product	PDSP Disposal cost				
[A]	[A1] 95	[A2] 45	[A3] 38		
[B]	[B1] 33	[B2] 212	[B3] 194	[B4] 27	[B5] 29

Table III.7 show the batch capacity of resources for each part; for example, the resource [RC4_1] can handle maximum 40 [A2] at one time; in other words, one [A2] occupies 2.5% (= 1/40) of the capacity of [RC4_1]. The occupation portion can be utilized for the batch capacity calculation of the different part type combinations; for example, 5 [B1] and 12 [B4] can be processed simultaneously by [RC5_2], while 5 [B1] and 13 [B4] cannot, because each case occupies 98% (= 0.1×5+0.04×12) and 102% (= 0.1×5+0.04×13) of the resource capacity respectively.

5. Cost for the operation processing, disposal, and delay penalty

The QRS scheduling requires a quality considered system performance measure which will be proposed in chapter VIII. The proposed measure requires the following cost related information:

- the operation processing and waiting cost per unit time;
- part disposal cost;
- remanufacturing delay penalty cost.

The disassembly operation cost by humans and machines are 2 and 1 per minute respectively, except for the case of using the specific tools in the workstation [WS3], the disassembly cost of which is 3.5 per minute. The costs for the refurbishing, cleaning, and reassembly are 2.5, 3, and 2 per minute respectively. The functional test costs 1 per minute. But the example QRS does not consider the waiting time cost, hence the operation waiting cost is 0 per minute.

The part disposal cost is set as 3 times bigger cost than the disassembly, cleaning, and refurbishing cost as in table III.8, which is based on the general statistics that remanufacturing can reduce PDSP remanufacturing cost by about 50-75% (Krill and Thurston 2005). The operation processing cost is calculated without consideration of the rework for simplicity only in this example, and the arrival lot size is selected as the number of simultaneously processed parts in batch resources. This thesis assumes the unlimited substitution of disposed parts (refer to assumption I.2), hence the disposal cost includes all the related cost for disposal and procuring a new corresponding parts, and so on. Appendix B discusses the operation processing cost derivation in detail.

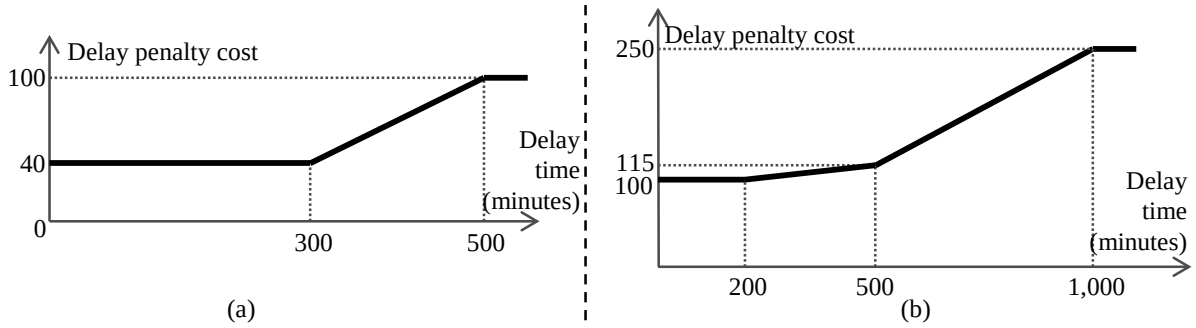


Figure III.6. Remanufacturing delay penalty cost graph for used products in the example QRS: (a) penalty for [A] and (b) penalty for [B].

The example QRS considers a delay penalty cost for the case the used products do not meet the due date. The penalty cost increases with the delay time, and the increase rate is different depending on the delay time range as in figure III.6; for example, used product [A]'s delay penalty calculation for the case of 200 and 400 minutes delay is different. The penalty cost between 0 to 300 minutes delay is always 40 independent of the delay time as shown in the graph in figure III.6(a), hence the penalty cost of the 200 minutes delay is 40. But the penalty cost for the delay between the 300 to 500 minutes is calculated by [delay time]×0.3-50 (refer to the range between 300 and 500 of the graph in figure III.6 (a)), hence the penalty of the 400 minutes delay is 70. The minimum delay penalty cost is arbitrarily set as about half of the total remanufacturing cost for both used product cases (refer to appendix B).

IV. QRS Quality Representation

Clarifying how quality is defined and handled is a prerequisite for the quality embedded remanufacturing system (QRS) simulation and control depending on the quality of used products and disassembled subassemblies/parts (PDSPs) and resources. Hence this chapter discusses the quality characteristics of the QRS and the quality representation method for the PDSP and resource.

Abbreviated terms used in this chapter:

BOL	Beginning Of Life;
EOL	End Of Life;
PDSP	used Product and Disassembled Subassembly/Part;
POF	Probability of Operation Failure;
POS	Probability of Operation Success;
QRS	Quality embedded Remanufacturing System;
RPQ	Ratio of PDSP output Qualities after an operation.

1. QRS quality definition

1.1. Quality characteristics of used products, subassemblies, and parts

1.1.1. Quality characteristics of products, subassemblies, and parts from the general perspective

The quality of products, subassemblies, and parts is usually dealt with in the manufacturing phase, where the manufacturer tries to achieve the predefined required specifications in the product design phase. The requirements of products, subassemblies, and parts are usually defined by two criteria: functional specifications and physical specifications. If one does not meet the specifications during manufacturing, it is regarded as a defective one; the quality is not satisfactory. In other words, the quality examination can be considered as the comparison of product, subassembly, and part realization with its required specification. Therefore this thesis defines quality of the product, subassembly, and part as follows:

"The degree of difference between the obtained and required specifications."

The quality examination method is not different among products/subassemblies/parts, while the characteristic of their specifications is different.

Mechanical parts have only physical specifications like dimension of width × length × height, diameter of a hole, mass, elasticity, and so on. Electrical or electromechanical parts like electronic chips have a little different specifications in addition; for example, insulation, inductance, electrical resistance, and so on.

Subassemblies have both physical and functional specifications. The functions of a subassembly are realized by joining composing parts as specified in the physical specifications of subassemblies (refer to figure IV.1). Therefore the physical specifications of a subassembly are specified by the dimensional and geometrical information of its joints and composing parts. In case of joining parts with joining elements like fixtures or fasteners, those joining elements also have physical specifications similar with that of parts like dimension and strength. Such functional specifications of a subassembly necessarily presume that the composing parts and joining elements fulfill their physical specifications. A subassembly's complete fulfilling the physical specifications logically results in satisfying all the required functional specifications of the subassembly. It means that the part arrangement requirement itself can be the functional specification of a subassembly.

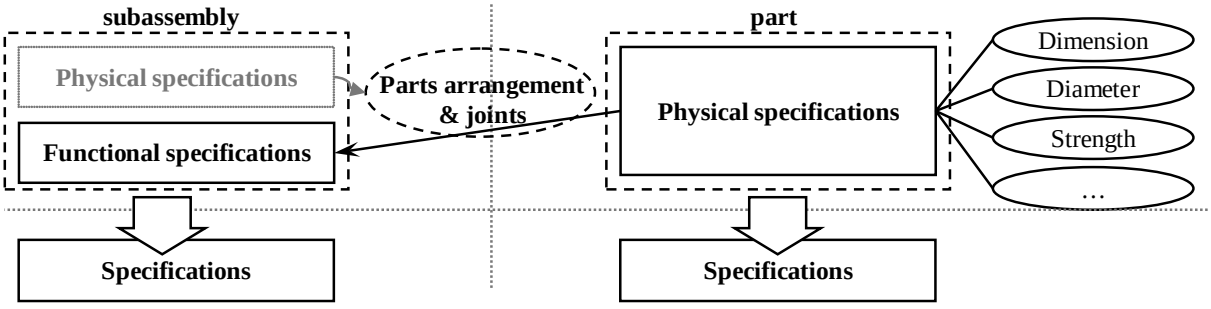


Figure IV.1. Relationships among subassemblies and parts from the general perspective.

Table IV.1. Difference between quality examination in a manufacturing system and QRS.

Characteristics	Target system	
	Manufacturing system	QRS
Final objective	– Decrease operation failure; – Change product design to decrease manufacturing difficulty.	– Maximize the system performance.
Necessary information	– Success rate of each operation; – Cause of the failure of an operation.	– Statistics of the operation processing time and POF depending on input part quality and resource performance.
Quality uncertainty recognition	– What to be decreased, and ultimately discarded.	– Necessary and natural characteristic.
Focusing quality characteristics	– Arrangement and joining of composing parts.	– Defects of joins among composing parts; – Recoverability of defects of parts.

This thesis classified above the subassembly specifications as functional and physical ones. But no definite conclusion on the classification criteria is drawn yet, because the classification can be different depending on the objective of information utilization; some publications distinguish physical and functional specifications (Kreng and Tseng-Pin 2004), but others do not (McKay *et al.* 2001 and Bourgeois *et al.* 2005). Although it is still controversial, this thesis considers an integrated approach. This thesis does not distinguish the type of specifications but integrates all requirements as just one specification type, because the important information for the use of quality examination results is not the classification but the effect of quality on the remanufacturing operations. Furthermore the specification classification can be different from domain to domain or even from product to product in the same domain. Hence this thesis considers it just as an integrated type to accept any kinds of specifications.

The relationship between a product and its composing subassemblies is similar with that of a subassembly and its composing parts, hence this thesis omits further explanations.

1.1.2. Quality characteristics of PDSP from the QRS perspective

The used product and disassembly subassembly/part (PDSP) quality examination method in the remanufacturing system is the same with that in the manufacturing system; a PDSP in the remanufacturing system which does not meet the required specifications is considered as unacceptable. Although the required level of specifications in the remanufacturing system can be a little different from that in the manufacturing system, this difference is not in the decision method but just in the specified values.

The above discussion on the quality specifications is highly biased from the assembly perspective. However this thesis assumes the reassembly as an always successful operation (refer to assumption I.4), and disassembly and refurbishment of disassembled parts are the main focus of the quality embedded remanufacturing system (QRS). Hence the objective of quality examination is different depending on the target system as summarized in table IV.1. The goal of quality examination in the manufacturing system is usually to decrease operation failure or to change the product design to decrease manufacturing difficulty. While the QRS in this thesis just examines

quality in order to maximize system performance. In other words, the QRS rather recognizes uncertain quality as an intrinsic natural characteristic, while the quality uncertainty in the manufacturing system is considered as what should be decreased and eliminated ultimately. This thesis does not discuss further on the other utilizations of quality examination in the QRS because they are out of this thesis's scope; for example, the quality can be examined for the design for disassembly, the resource quality enhancement, the guide to prevent the product misuse, and so on.

This thesis utilizes historical statistical information for the QRS performance maximization: the operation processing time and the probability of operation failure (POF) depending on the input PDSP quality and the performance of resources in charge of the operation. The effect of resources on operations and the operation failure cases will be discussed in the following sections IV.1.2 and IV.2 respectively. Since the operation processing time and POF can be different depending on the input quality, the statistical information on the operation processing should be managed with classification not only by operations but also by the input PDSP quality. The QRS utilizes this information to select the best remanufacturing process for each PDSP as well as the best dispatching rule for each resource depending on the system states. The scheduling mechanism utilizing the quality dependent statistic information will be discussed in chapter VIII.

1.2. Quality characteristics of resources

The operation processing time and POF are affected not only by the input PDSP quality but also by the resource performance which does the operation. A resource can do an operation very well but do another operation very badly depending on its familiarity or structural suitability to the operations. The performance of human resources can be different depending on their expertise for the operation, using tools, and so on, and machine resources show different performance depending on their types, accuracy, and so on.

Each resource has its own maintenance criteria; for example, a machine should maintain its motor speed, the tool alignment precision, and so on. Their states transit in relatively short time depending on the degree of fatigue or exceptional events like a tool break, and such characteristics can also affect the resource performance. But this thesis assumes stable resource performance (refer to assumption I.6) to decrease problem complexity and does not deal with such resource isolated characteristics, which should rather be controlled at a lower level by the resource itself. Hence what this thesis should consider is that resources do not show similar performance to different operations, and the resource quality is defined as follows:

"The relative performance for an operation to the average performance of all resources which can process the operation."

Resource performance means the operation processing time and POF as defined in chapter I. A resource showing faster processing time and a lower POF is considered as having better quality. A resource can also show faster processing time but higher POF, or vice versa. Hence this thesis permits the directions of two performance criteria to be decoupled.

2. Representation of quality related statistical information

The focus of the QRS is not the quality itself but the quality dependent statistical information: the operation processing time and POF. The failure of operations can be decided by two criteria:

i) the completion or not of the operation, and
ii) the additional PDSP defects occurrence during the operation.

Operation completion means that the objective of the operation was achieved; for example, the completion of a subassembly disassembly operation corresponds to the separation of the subassembly into its composing subassemblies or parts (refer to figure IV.2, the variables like $pc_{o,iq}$ and $RPQ^{nc}{}_{o,iq}$ are explained later in section IV.2.1). Hence this research classifies the operation success or failure into four cases by combining the above two criteria:

– non-completion without additional defects;
– non-completion with additional defects;
– completion with additional defects;
– completion without additional defects.

The first three are failure cases and the last one is the success case. The additional defects mean only the directly occurred defects during the operation; in other words, the case of hidden defects exposure by disassembly is not considered as defects occurrence during the operation in this thesis. Hence the hidden defect exposure cases are also classified as success cases.

Although we discussed that the resource quality also affects the difference of the operation processing time and POF, considering the resource quality simultaneously with the input PDSP quality makes the information dimension too complex. Hence this thesis represents input PDSP quality dependent statistical information as the main information, and the resource quality effect as a supplementary correction factor.

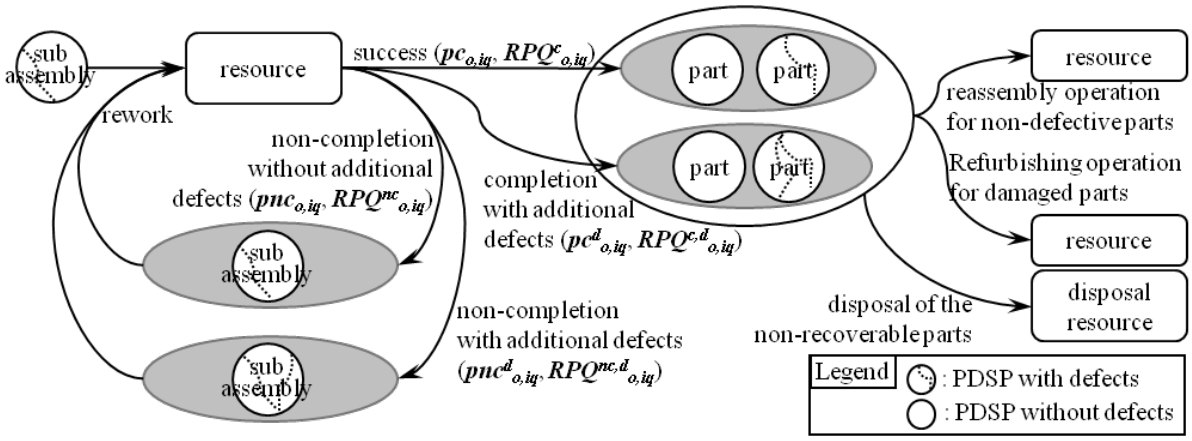


Figure IV.2. Example of disassembly results and following actions depending on the results.

2.1. Statistical information depending on the input PDSP quality

This thesis focuses on not the PDSP quality itself (refer to appendix C) but the statistical information depending on the PDSP quality as discussed in section IV.1.1.2. Hence this research considers the quality $\boldsymbol{q_p}$ of a PDSP $\boldsymbol{p}$ as a given value, which is represented as a discrete variable as follows:

$\boldsymbol{q_p} \in \boldsymbol{Q_p} = \{\boldsymbol{q_{p,1}}, \boldsymbol{q_{p,2}}, \ldots\}$,

where

$\boldsymbol{Q_p}$ set of possible qualities of the PDSP $\boldsymbol{p}$; (Def. IV.2.1)

$\boldsymbol{q_{p,i}}$ quality instance.

$\boldsymbol{Q_p}$ of a PDSP varies from PDSP to PDSP. It can be also different depending on modelers even for the same PDSP. This thesis defines no further restrictions but suggests a simple guideline:

– make $\boldsymbol{Q_p}$ as simple as possible;
– define qualities showing a different operation processing time, POF, or process as separated quality instances.

The statistical information of operations processing is handled on two levels (refer to table IV.2):

–
– 1st level: the operation processing time and the probabilities of the three operation failure cases (POF);
– 2nd level: the ratio of PDSP output qualities after an operation (RPQ) for each operation success and failure case.
–

The first level information $\boldsymbol{tf_{o,iq}}$ for the combination of a PDSP $\boldsymbol{p}$'s operation $\boldsymbol{o}$ and the input quality $\boldsymbol{iq} \in \boldsymbol{Q_p}$ can be simply defined as follows:

$\boldsymbol{tf_{o,iq}} = (\boldsymbol{pt_{o,iq}}, \boldsymbol{pnc_{o,iq}}, \boldsymbol{pnc^{d}_{o,iq}}, \boldsymbol{pc^{d}_{o,iq}})$, (Def. IV.2.2)

where

$\boldsymbol{pt_{o,iq}} \in \mathbf{F}^{\mathbf{PDF}}$ statistical information of operation processing time; (Def. IV.2.3)

Table IV.2. Statistical information representation on PDSP output qualities of an operation.

Statistical information			Definition	
			Level 1	Level 2
Output PDSP quality	Operation non-completion	non-additional defects	$\boldsymbol{pnc_{o,iq}}$	$\boldsymbol{RPQ^{nc}_{o,iq}}$
		additional defects	$\boldsymbol{pnc^{d}_{o,iq}}$	$\boldsymbol{RPQ^{nc,d}_{o,iq}}$
	Operation completion	additional defects	$\boldsymbol{pc^{d}_{o,iq}}$	$\boldsymbol{RPQ^{c,d}_{o,iq}}$
		non-additional defects	$\boldsymbol{pc_{o,iq}}$	$\boldsymbol{RPQ^{c}_{o,iq}}$

$pnc_{o,iq} \in \mathbb{R}^{0+}$ probability of operation non-completion without additional defects; (Def. IV.2.4)

$pnc^{d}_{o,iq} \in \mathbb{R}^{0+}$ probability of operation non-completion with additional defects; (Def. IV.2.5)

$pc^{d}_{o,iq} \in \mathbb{R}^{0+}$ probability of operation completion with additional defects. (Def. IV.2.6)

The sum of $pnc_{o,iq}$, $pnc^{d}_{o,iq}$, and $pc^{d}_{o,iq}$ can be defined as POF as follows:

$$pof_{o,iq} = pnc_{o,iq} + pnc^{d}_{o,iq} + pc^{d}_{o,iq}, \quad \text{(Def. IV.2.7)}$$

which should not exceed 1. The remainder of $pof_{o,iq}$ to 1 is the probability of operation success (POS) which is defined as follows:

$$pc_{o,iq} = 1 - pof_{o,iq}. \quad \text{(Def. IV.2.8)}$$

The operation time statistics $pt_{o,iq}$ has the form of a statistical distribution, because the operation processing time is usually different at every trial. Although some batch operations like cleaning may show always exactly the same processing time, it can be also expressed by a deterministic distribution.

The second level is the conditional information for each operation success and failure case in figure IV.2. It represents the quality change of a PDSP $\boldsymbol{p}$ by additional defect occurrences or hidden defect exposures. The RPQ information $RPQ^{nc}_{o,iq}$ of operation non-completion without additional defects case can be defined as follows:

$$RPQ^{nc}_{o,iq} = \{(oq, rto^{nc}_{o,iq,oq}) \mid oq \in Q_p, rto^{nc}_{o,iq,oq} \in \mathbb{R}^{0+} \leq 1, \text{ and } \Sigma_{oq} rto^{nc}_{o,iq,oq} = 1\}, \quad \text{(Def. IV.2.9)}$$

where

oq possible output quality of the input PDSP $\boldsymbol{p}$ after an operation $\boldsymbol{o}$ for the given input quality $\boldsymbol{iq}$;

$rto^{nc}_{o,iq,oq}$ occurrence frequency ratio of the output quality $\boldsymbol{oq}$. (Def. IV.2.10)

The set should contain all possible qualities of the operation non-completion without additional defects case after the operation. Hence the sum of the output quality occurrence ratios for the given operation and input quality should be 1 as included in the condition of the sets: $\Sigma_{oq} rto^{nc}_{o,iq,oq} = 1$. The RPQ information for other cases is defined as follows:

$RPQ^{nc,d}_{o,iq}$ RPQ information for the operation non-completion with additional defects case; (Def. IV.2.11)

$RPQ^{c,d}_{o,iq}$ RPQ information for the operation completion with additional defects case; (Def. IV.2.12)

$RPQ^{c}_{o,iq}$ RPQ information for the operation completion without additional defects case.[1] (Def. IV.2.13)

The detailed definitions are omitted because they have exactly the same form with $RPQ^{nc}_{o,iq}$.

[1] In this case, there could be more than one output quality since hidden defects could be uncovered by the disassembly operation.

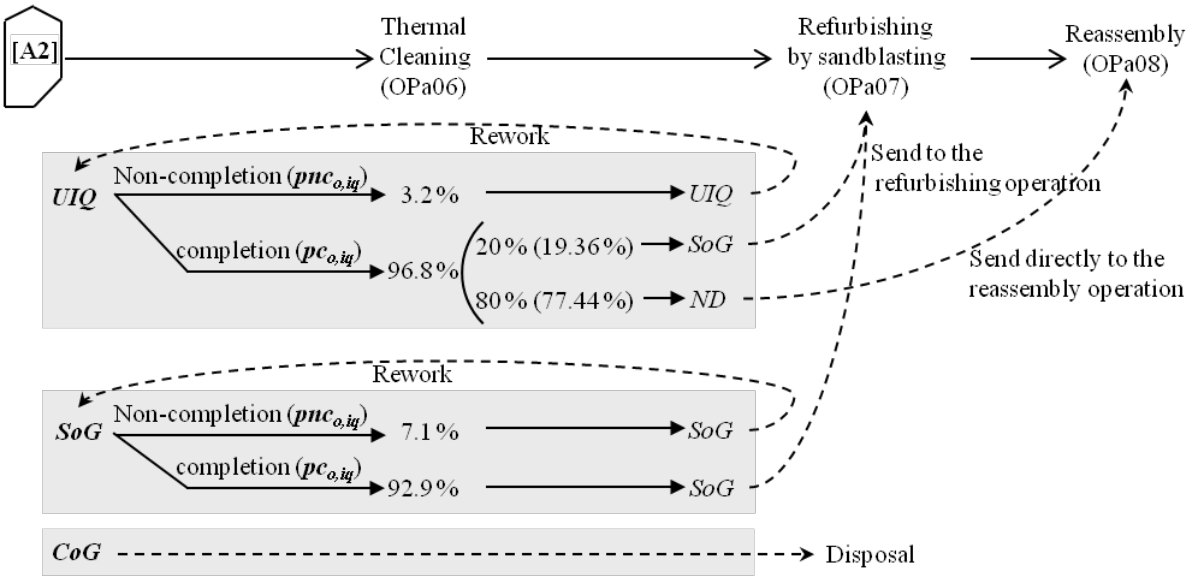


Figure IV.3. Part [A2] quality change by the operation OPa06 in the example QRS.

While the possible output quality for the operation non-completion cases is in the PDSP ***p***'s possible quality set $\boldsymbol{Q_p}$ without exception, the possible output quality for the operation completion cases of disassembly operations is more complicated. The disassembly completion means a PDSP is separated into some composing PDSPs. Therefore the quality of each separated PDSP should be indicated, and such indication highly increases the representation complexity. This thesis adopts an indirect way to reduce the complexity. The RPQ for the operation completion case of disassembly operations is also defined as the same with that of operation non-completion case; the output quality of disassembly operation completion case is also in $\boldsymbol{Q_p}$ of the input PDSP ***p***. The quality information of a subassembly incorporates the quality of all composing parts and their joints, hence there is no information loss. Consequently a subassembly quality can be easily converted into the disassembled PDSPs' quality by discarding the quality information of the disconnected joints after disassembly, since the joint does not exist anymore. Hence this research always specifies the operation output as one value and supports a mapping function $\boldsymbol{\lambda^{cq}}$ (def. VI.2.24) between the PDSP quality and its composing PDSPs qualities, which will be defined in chapter VI.

This thesis shows the quality and statistical information representation for the part [A2] cleaning operation OPa06 of the example QRS in chapter III to assist the explanation of the above defined quality and statistical information representation. Figure IV.3 represents the possible quality of the disassembled [A2] and its quality change by OPa06. The possible quality set of [A2] during its processing can be defined as $\boldsymbol{Q_{[A2]}}$ = {*UIQ*, *ND*, *CoG*, *SoG*}; the meaning of each element of this set is as follows (refer to section III.2.1):

UIQ unidentified;
ND no defects;
CoG crack or strong wear of the grooves;
SoG surface scratch or weak wear of the grooves.

A disassembled [A2] can have the initial quality *UIQ*, *SoG*, or *CoG* right after disassembly. But its quality

cannot be assigned as *ND*, because it may have surface scratch or weak wear of the grooves which may be hidden by dirt. The quality of [A2] having the *UIQ* quality can be clarified by OPa06 as one of *ND* or *SoG*. While the *SoG* quality [A2] after OPa06 should be refurbished, the *ND* quality [A2] is eligible to be reassembled. Hence the *ND* quality [A2] skips the refurbishing operation OPa07. The *CoG* quality [A2] is directly disposed of, because the defect cannot be recovered by cleaning or refurbishing operations. Hence the statistical operation processing information for OPa06 is required for the following input quality cases: *UIQ* and *SoG*. If we assume that 20% of the successfully cleaned *UIQ* quality [A2] by OPa06 shows scratches on its surface, the processing time and POF information of the case can be modeled as $\boldsymbol{tf}_{OPa06,UIQ}$ = (Deterministic(16.2), 0.032, 0, 0) and $\boldsymbol{RPQ}^{c}_{o,iq}$ = {(*ND*, 0.8), (*SoG*, 0.2)}. The first element of $\boldsymbol{tf}_{OPa06,UIQ}$ indicates its processing time distribution of OPa06 (refer to table III.4), and the other three elements of it represent the occurrence probabilities of the three kinds of failure cases (refer to figure IV.2); 0.032 and the two zeros mean that 3.2% of [A2] are not cleaned correctly and no additional defects are generated by OPa06 respectively.[1] (*SoG*, 0.2) in $\boldsymbol{RPQ}^{c}_{o,iq}$ means that 20% of the operation success case result in *SoG*. Hence the operation OPa06 with the input quality *UIQ* results in 3.2% of rework, 19.36% (= (1-0.032)×0.2) of *SoG* which will be sandblasted in the next operation OPa07, and 77.44% (= (1-0.032)×0.8) of *ND* which will be directly reassembled in the reassembly operation OPa08 without sandblasting. Appendix E represents the statistical information for the other PDSP and operation combinations in detail.

The modeler should be cautious that the output quality of an operation is not exclusive to the four kinds of operation results in figure IV.2; the same quality can result in both the operation completion case and non-completion case like *SoG* after OPa06, because operation completion only focuses on the operation objective. Hence modelers do not have to define the same quality state differently for the operation completion and non-completion cases.

2.1.1. Correlation between the operation time and output PDSP quality

The operation processing time is represented with well known statistical distributions like normal, uniform, exponential distributions. But this thesis represents an operation processing time not just by one statistical distribution but by a statistical distribution which has correlation factors depending on the output PDSP quality, because the operation processing time can have different distributions depending on the output PDSP quality; for example, in case the head of a bolt joining two parts is worn away during the disassembly operation, the disassembly can fail and the operation time increases, on the contrary, a part crack can cause early completion of the disassembly operation. Although a correlation embedded statistical distribution in one formula is preferable, constructing such formula is difficult; because the PDSP quality has non-numeric characteristics, and the operation failure does not have the same directional effect on the operation processing time but sometimes opposite directional effects depending on each case.

This thesis suggests an alternative representation of the correlation factors by gathering two kinds of information: the operation processing time distribution for a representative output PDSP quality case, and statistical distribution shape transformation factors for each combination case of the PDSP input and output

[1] OPa06 is assumed to cause no additional defects; the *SoG* after OPa06 is not caused (but exposed) by the operation.

qualities and operation completion or not. This thesis sets the operation success case as the representative, and considers correction factors for the distribution parameters as shape transformation factors. Hence the processing time distribution $\boldsymbol{pt_{o,iq}}$ (def. IV.2.3) in the operation processing time and POF information $\boldsymbol{tf_{o,iq}}$ (def. IV.2.2) is for the operation success case $\boldsymbol{pc_{o,iq}}$ (def. IV.2.8), and correlation information should be supported for the other cases showing different processing time. The correlation set $\boldsymbol{CR_o}$ of operation $\boldsymbol{o}$ is defined as follows:

$$\boldsymbol{CR_o} = \{(\boldsymbol{iq}, \boldsymbol{oq}, \boldsymbol{cmp}, \boldsymbol{cf^{c\text{-}pt}_{iq,oq,cmp}}) \mid \boldsymbol{iq},\boldsymbol{oq} \in \boldsymbol{Q_p}, \boldsymbol{cmp} \in \mathbb{B}, \text{ and } \boldsymbol{cf^{c\text{-}pt}_{iq,oq,cmp}} \in \mathbb{R}^{0+}\}, \quad \text{(Def. IV.2.14)}$$

where

$\boldsymbol{cf^{c\text{-}pt}_{iq,oq,cmp}}$	correlation correction factor to the processing time for the case of ***iq***, ***oq***, and ***cmp***;	(Def. IV.2.15)
iq	input PDSP quality;	
oq	output PDSP quality;	
cmp	operation completion of not.	

Although the correction factor to each parameter of $\boldsymbol{pt_{o,iq}}$ can be different and even the distribution type can be different in some cases, the difference is presumed not too much. Hence this thesis reduces the complexity with the following assumptions (refer to assumption I.8):

– the distribution type of $\boldsymbol{pt_{o,iq}}$ is the same independent of the output quality,
– and the correlation factor for each parameter of $\boldsymbol{pt_{o,iq}}$ is also same.

Consequently each instance in $\boldsymbol{CR_o}$ requires only one transformation factor $\boldsymbol{cf^{c\text{-}pt}_{iq,oq,cmp}}$ which will be multiplied with each distribution parameter of the corresponding operation processing time statistics $\boldsymbol{pt_{o,iq}}$ (def. IV.2.3); for example, 20% shorter processing time in the case of [A2] cracking during the refurbishing operation OPa07 of the example QRS in chapter III can be represented as $\boldsymbol{CR_{OPa07}} = \{(SoG, CoG, *, 0.8)\}$[1]. Hence the OPa07 processing time distribution $\boldsymbol{tf_{OPa07,SoG}}$ is shifted from Normal(4.4, 0.4) (refer to table III.4) to Normal(3.52, 0.32) by multiplying each parameter of $\boldsymbol{tf_{OPa07,SoG}}$ with 0.8.

2.2. Statistical information depending on the resource quality

The resource quality is different from operation to operation as discussed in section IV.1.2, therefore it should be specified to the combination of a resource and an operation. The resource quality is the effect of a resource ***r*** on the processing time and POF of an operation ***o***. In other words, the effect of a resource quality can be considered as the adjustment of statistical information on the operation processing time and POF $\boldsymbol{tf_{o,iq}}$ (def. IV.2.2) by the resource quality correction factors $\boldsymbol{rq_{o,r}}$ which is defined as follows:

$$\boldsymbol{rq_{o,r}} = (\boldsymbol{cf^{r\text{-}pt}_{o,r}}, \boldsymbol{cf^{r\text{-}POF}_{o,r}}), \quad \text{(Def. IV.2.16)}$$

[1] *SoG* and *CoG* correspond to the [A2] quality of the surface scratch or weak wear of the grooves and the crack or strong wear of the grooves, respectively. The asterisk * represents all possible values: true and false, hence * in the third column means both the operation completion and non-completion cases.

where

$cf^{r\text{-}pt}_{o,r} \in \mathbb{R}^{0+}$ processing time quality of ***r***; relative performance on the processing time of ***o***; (Def. IV.2.17)

$cf^{r\text{-}POF}_{o,r} \in \mathbb{R}^{+}$ POF quality of ***r***; relative performance on the POF of ***o***. (Def. IV.2.18)

The $cf^{r\text{-}pt}_{o,r}$ factor represents ***r***'s processing time relative to the average processing time by all capable resources for the operation ***o***. The $rq_{o,r}$ definition adopts similar assumptions with that applied to the operation processing time and input/output quality correlation factor $cf^{c\text{-}pt}_{iq,oq,cmp}$ (def. IV.2.15) definition; we assume that the resource quality correction factors to the parameters of an operation processing time distribution $pt_{o,iq}$ (def. IV.2.3) are the same. In the same way, $cf^{r\text{-}POF}_{o,r}$ is presumed to be similar for all three kinds of failure cases in figure IV.2. Hence only one value is defined for each effect on the operation processing time and POF to reduce the complexity. The resource performance reflected operation processing time and POF information $tf_{o,iq,r}$ can be derived with $tf_{o,iq}$ (def. IV.2.2) and $rq_{o,r}$ as follows:

$$tf_{o,iq,r} = (pt_{o,iq} \times cf^{r\text{-}pt}_{o,r},\ pnc_{o,iq} \times cf^{r\text{-}POF}_{o,r},\ pnc^{d}_{o,iq} \times cf^{r\text{-}POF}_{o,r},\ pc^{d}_{o,iq} \times cf^{r\text{-}POF}_{o,r}).$$

Here $pt_{o,iq} \times cf^{r\text{-}pt}_{o,r}$ means multiplying $cf^{r\text{-}pt}_{o,r}$ to each PDF parameter of $pt_{o,iq}$; for example, the quality of the two resources RC5_1 and RC5_2 in the workstation WS5 for the operation OPa06 in the example QRS in chapter III can be represented as $rq_{OPa06,RC5\_1}$ = (1.10, 1.0) and $rq_{OPa06,RC5\_2}$ = (0.93, 0.91) respectively (refer to table III.6). Then the statistical information of OPa06 by [RC5_2] for the input quality *UIQ* is transformed from $tf_{OPa06,UIQ}$ = (Deterministic(16.2), 0.032, 0, 0) (refer to table III.4) to $tf_{OPa06,UIQ,RC5\_2}$ = (Deterministic(15.1), 0.029, 0, 0), where 15.1 and 0.029 are calculated by 16.2×0.93 and 0.032×0.91 respectively.

V. Multi-agent Approach for the QRS

This chapter presents how to apply the multi-agent approach to the quality embedded remanufacturing system (QRS), which is an overall framework from the target remanufacturing system modeling to the constructed system control. The framework also includes a multi-agent structure which is the basis of the proposed modeling tools and the real-time scheduling mechanism described in the following chapters. All the agents in the structure are autonomous and pursue their own objectives through communication with other agents. Chapter VIII will discuss the objective of each agent and the communication among agents in detail.

Abbreviated terms used in this chapter:

DTPN	Dynamic Token Petri-Net;
IRSR	Intuitive Remanufacturing System Representation;
PDSP	used Product and Disassembled Subassembly/Part;
QRS	Quality embedded Remanufacturing System;
RFID	Radio Frequency IDentification;
XCPN	eXtended two-level Colored Petri-Nets.

1. Multi-agent approach framework for the QRS

1.1. Overall framework

The QRS in this research can be divided into two parts: the real-world remanufacturing system and the corresponding virtual remanufacturing system (refer to figure V.1). The latter one can be rather called a software system for the real-world system control. The real-world remanufacturing system should support the following requirements for the application of a multi-agent approach:

- The examined quality of used products and disassembled subassemblies and parts (PDSPs) in the real-world should be immediately announced to the corresponding PDSP agents in the virtual system;
- All elements including PDSPs in the remanufacturing system should be able to communicate with the virtual system to accept and follow the execution orders from the virtual system.

For the first requirement, the PDSP quality should be examined after every operation, and the result should be sent to the virtual system. A machine or human may examine visually or with some inspection tools and transmit the quality information to the virtual system. The wired/wirelessly linked inspection equipment to the virtual system enable the direct transmission of the examined quality information to the real-world control system, while the quality information examined by humans should be entered manually through a terminal linked to the virtual system at each examination point. The human can also alternatively use a personal digital assistant or its compatible equipment which has wireless communication functions.

The second requirement can be realized by attaching required sensors and wireless communication devices like radio frequency identification (RFID) tags to each PDSP and resource. Resources do not necessarily need wireless functions, because they are usually fixed at certain positions. Although human resources can move, they usually work at certain fixed places. Hence wired communication functions are enough for resources to realize the multi-agent quality embedded remanufacturing system (QRS). Attaching sensors of wireless function chips to each PDSP can be difficult for the small sized or complex structured PDSPs. This thesis selects a container utilization approach, where the remanufacturing system controls containers which have sensors and wireless chips and contain PDSPs. Utilizing containers may cause a redundant processing time by saving/retrieving PDSPs to/from containers. But this thesis does not consider such additional processing time, because such handing time is probably small compared to the processing time of the disassembly, refurbishing, and so on.

The remanufacturing system equipped with the above mentioned prerequisite functions can be controlled by the following four steps which following subsections will consecutively discuss in detail (refer to figure V.1):

- Step 1: model the real-world remanufacturing system with modeling tools which a virtual system can interpret;
- Step 2: construct the corresponding virtual multi-agent system and load detailed information to each agent as modeled in step 1;
- Step 3: acquire knowledge for the scheduling mechanism embedded in the virtual system by analyzing the

model with a simulation approach;

– Step 4: synchronize the virtual system to the real-world remanufacturing system and execute it[1] with the embedded real-time knowledge-based scheduling mechanism.

1.2. Detailed steps for the multi-agent approach application to the QRS

1.2.1. Step 1: Modeling the real-world remanufacturing system

Although modeling of the remanufacturing system necessarily requires proper modeling tools which can represent the required information for the QRS control and simulation, no appropriate tools are found among the previous related research (refer to section II.1). Hence this thesis proposes new modeling tools. Clarifying the necessary information for the QRS control and simulation is mandatory to develop proper QRS modeling tools, and this thesis inspected the general remanufacturing systems and extracted the necessary information. Section VI.1 will discuss the required information in detail.

The proposed QRS modeling tools comprise a user-side modeling tool and a system-side modeling tool, and the modeler will start to represent the remanufacturing system with the user-side modeling tool. The system-side model is unsuitable for the model verification, because the model contains detailed system level information like a multi-agent structure or agent states, and such information is unfamiliar and difficult to understand for remanufacturing engineers who are basically interested in the field knowledge in remanufacturing. Hence an intuitive and easy-to-understand user-side model is required, which can decrease the difficulty in modeling and interpreting.

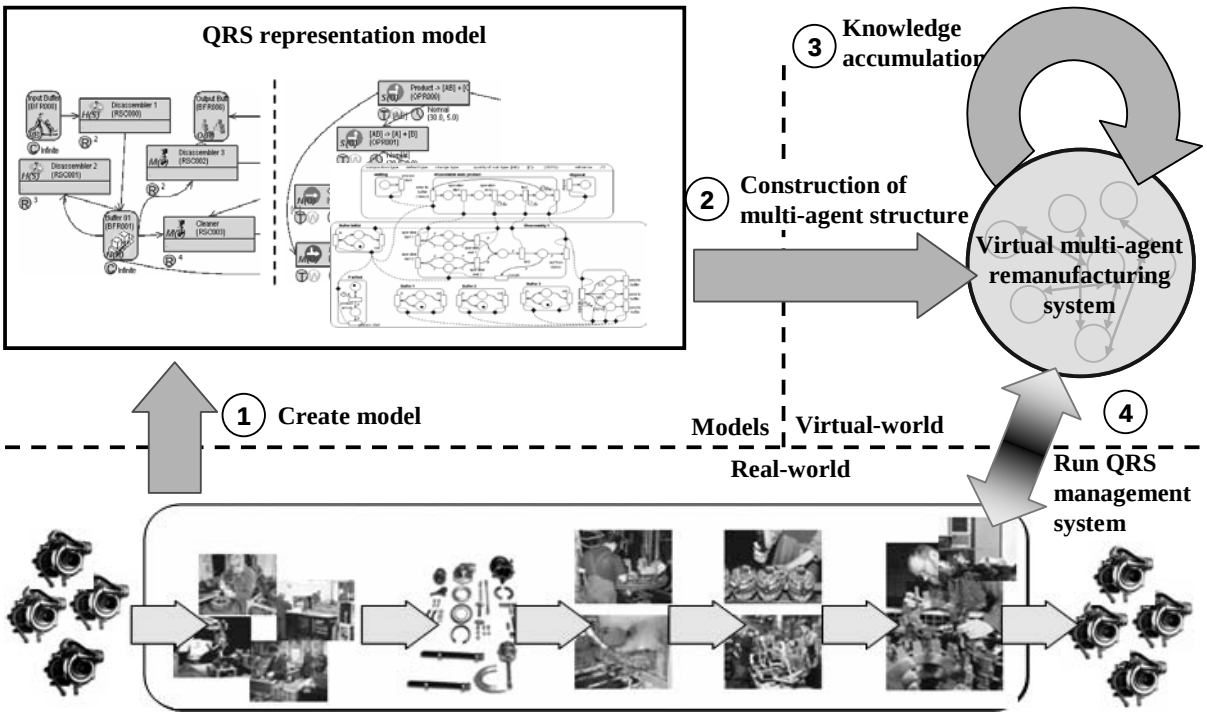


Figure V.1. Multi-agent approach framework for the QRS.

[1] This is out of the scope of this thesis, hence no further discussions are in the following chapters.

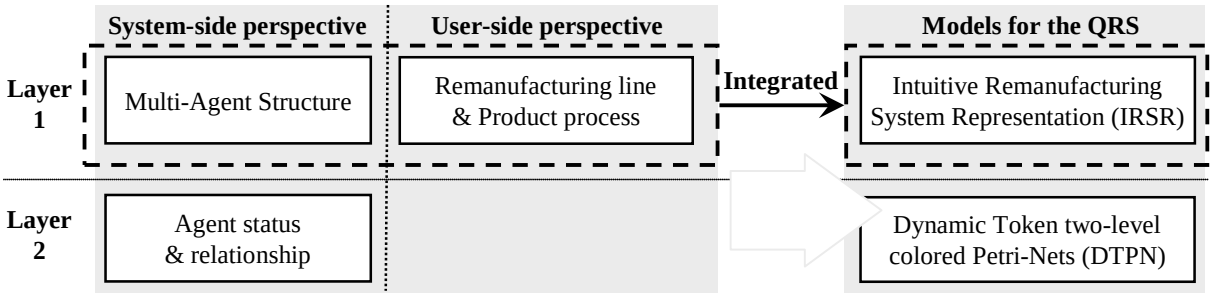


Figure V.2. Two layered model for a QRS representation.

The system-side model is constructed based on a built user-side model to represent the necessary information for the system control. The system-side model can be designed with two layers. The upper layer represents the overall structure of the multi-agent system, and the lower layer contains detailed agent information: possible agent states and their relationships. Although the information in two layers can be integrated into one model together, the integrated one is so complex to clarify what has to be defined as agents. Therefore the two layered modeling approach is more reasonable to construct the multi-agent virtual system.

Although constructing a sound virtual system requires three models: one user-side model and two system-side models, the overall multi-agent structure for the virtual system can be easily extracted from the user-side model. Hence this thesis integrates the user-side model and the system-side first layer model (refer to figure V.2). This thesis also developed a conversion method from the integrated model to the system-side second layer model to avoid inconsistency between them. The latter model should contain additional information which does not exist in the integrated model: possible agent states, a synchronization protocol among agents, and so on. The developed conversion method provides such information. Hence the conversion in this thesis means not only the change of the expression but also the addition of pre-acquired information; for example, a resource agent has the states of waiting operation request, processing operations, examining output PDSP quality, and so on.

This thesis proposes two QRS modeling tools which handle the above requirements: the intuitive remanufacturing system representation (IRSR) and the dynamic token two-level colored Petri-nets (DTPN). IRSR represents the user-side model and the overall multi-agent structure. The IRSR model can be utilized not only for representing remanufacturing shop facilities and PDSPs of the QRS but also for extracting the required agents in the virtual multi-agent system. DTPN represents the system-side second layer model, which contains agents' internal states information and their relationships. A conversion method from an IRSR model into a corresponding DTPN model is also proposed. Although there can be some tacit general patterns in models designed by the people who know two-level Petri-nets and remanufacturing systems well, it does not guarantee the consistency among the designed QRS models. Various QRS models can be obtained, if they are designed by field technicians. Extracting the required information from such various QRS models requires very complicated interpreter software. Consequently the conversion method is not an optional element for the proposed modeling tools but a necessity. Chapter VI will explain the modeling tools and the conversion method in detail.

1.2.2. Step 2: Constructing the corresponding virtual multi-agent system

The virtual multi-agent structure corresponding to the real-world remanufacturing system can be created based

on the IRSR and DTPN models created in step 1. The virtual system can be executed by embodying interpretation and utilization logic, which will be discussed in chapter VIII, for the modeled information to each agent, because the IRSR and DTPN models only contain the multi-agent structure, agent relationships, and statistical information.

The virtual multi-agent system is constructed in two steps: creating agents and loading necessary information on each agent. The IRSR model is utilized for the agent creation; each resource, workstation, buffer and PDSP in the IRSR model is created as an agent. Each created agent loads the corresponding information from the DTPN model, and utilizes the loaded information by the embedded execution logic for its state control and synchronization with other agents. Section V.2 of this chapter will explain the possible agent states, state transitions, and agent relationships in detail. The embedded logic in each agent also encloses the functions which can interpret static and dynamic information of corresponding PDSPs and resources. The main logic of PDSP agents is the operation and resource selection method, while that of resource agents is the PDSPs ranking method under the allocated dispatching rule. Hence PDSP/resource agents interpret supported information and utilize it for the remanufacturing process selection and PDSP dispatching sequence decision respectively. Chapter VIII will discuss the execution logic on the operation/resource selection and the dispatching rule allocation in detail.

The created agents do their work based on the information loaded inside them. But outside stimulations are sometimes required to transit their states; for example, changing a part state from the ‘waiting operation’ to the ‘under processing’. Those stimulations are received from two kinds of sources: other agents or corresponding real-world elements. Once a PDSP agent selected an operation and a resource to do the operation, the start and end of the operation processing is dependent on the selected resource’s actions. Hence the PDSP’s state transits depending on the processing start and end messages from the resource agent. On the other side, the resource state change is dependent on the actions of the corresponding real-world resource; for example, the operation processing end of a real-world resource is reflected to the corresponding resource agent in the virtual system by their synchronization.

1.2.3. Step 3: Accumulating scheduling knowledge

The proposed QRS scheduling mechanism adopts a knowledge-based approach. Hence the necessary knowledge should be accumulated in the virtual system before execution. The knowledge in the proposed scheduling mechanism corresponds to the relationship between the best dispatching rule for each resource and the system states (refer to figure V.3). The simulation right at the required time obviously produces better results, because it uses exact information of the system state. But the simulation requires considerable computation time and produces too old simulation results to apply to the real-time system. Consequently applying such lagged results cannot satisfy the expected system performance, because the system state at the application time will be quite different from the analyzed system state. Hence this thesis considers a knowledge-based approach with off-line simulations. But the simulations for all combinations of QRS system states and dispatching rules of each resource also require tremendous computation time. Therefore some heuristic methods assist the knowledge accumulation to reduce the simulation time. Sections VIII.6.1 and VIII.6.3 will respectively explain the knowledge and the heuristic knowledge accumulation in detail.

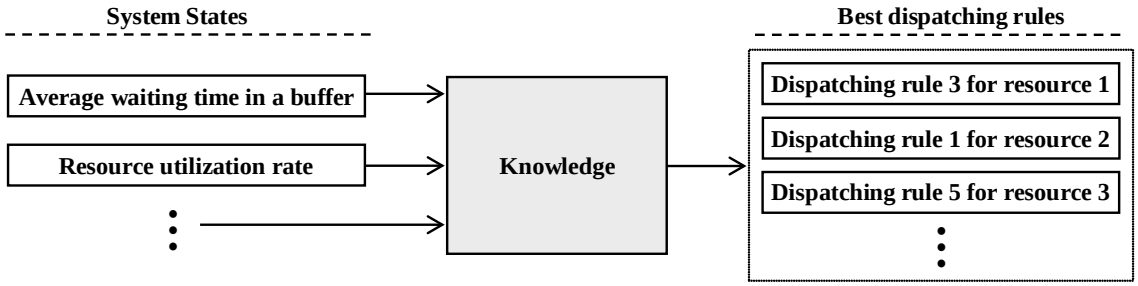


Figure V.3. Knowledge in the QRS scheduling mechanism.

Anything representing the QRS can be defined as system states; for example, if PDSPs flow well in a remanufacturing shop, if machines are busy, if there is any bottleneck, if there are too many PDSPs on the shop, and so on. Those states are represented by some state variables like the average waiting time in buffers, the resource utilization rate, and so on. The simulations for knowledge accumulation should be conducted for all possible values of state variables to cover all possible system states. Therefore the continuous variables should be converted into discrete variables by dividing the possible values into a discrete set of ranges, because the simulation for each value case is impossible for the continuous variables. Section VIII.6.2 discusses the system state variables in detail.

1.2.4. Step 4: Executing a QRS with a real-time scheduling mechanism

The real-world remanufacturing system can be executed by synchronization of each agent in the virtual system to the corresponding element in the real-world system. Agents in the virtual system maintain real-time information by synchronization, and decide their actions in accordance with the embedded logic, system states, and communication results with other agents. Agent decisions are announced to the real-world remanufacturing system which exactly follows the execution orders from the virtual system. The real-world system also feeds some required information for the decision and state transition of agents in the virtual system; quality examination results, operation processing end messages, and so on.

The virtual multi-agent system in the proposed QRS framework makes all decisions for the QRS simulation and control of the real-world system. Hence the virtual multi-agent system embeds the scheduling mechanism which contains a communication language and negotiation mechanisms. Agents negotiate with each other to pursue their own objectives which should be consistent to the global objective of the system. Hence the virtual system should specify a common performance measure, and prompt the agents to pursue minimizing or maximizing the measure. Chapter VIII will explain the communication protocol and scheduling mechanism in detail.

2. Multi-agent structure for the QRS

2.1. Agent types in the multi-agent structure

The agents in the proposed multi-agent structure can be classified into four groups: the PDSP agent group, facility agent group, PDSP life management agent group, knowledge support agent group (refer to figure V.4).

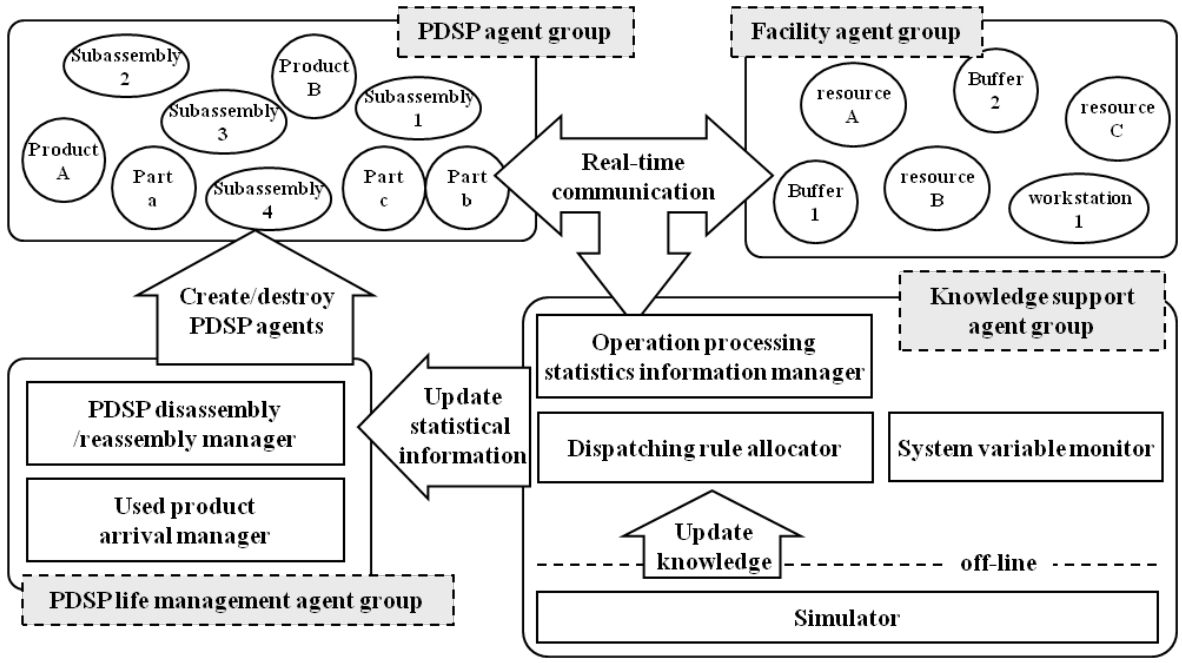


Figure V.4. Multi-agent structure classified into four groups.

The agents in the PDSP and facility agent groups are created based on the IRSR model and contain corresponding DTPN modules for the states handling. The other agents in the PDSP life management and knowledge support agent groups are basic agents independent of the modeled information. All agents except for the agents in the PDSP life management group communicate with each other for their decision making to maximize system performances. Sections VI.4 and VIII.3.2 will explain respectively the each agent's corresponding DTPN modules and the communication among agents in detail (refer to figures VIII.1 - 4).

2.1.1. PDSP agent group

All PDSP agents belong to this group. Each PDSP can have the following states: 1. just created, 2. deciding next action, 3. collecting system states information, 4. selecting operation/resource, 5. waiting for operation processing, 6. under processing, 7. examining quality, and 8. ending life. Figure V.5 shows the detailed state transition diagram[1] of the PDSP agent. The detailed explanation on the state transitions by communication with other agents will be explained in section VIII.3.2.

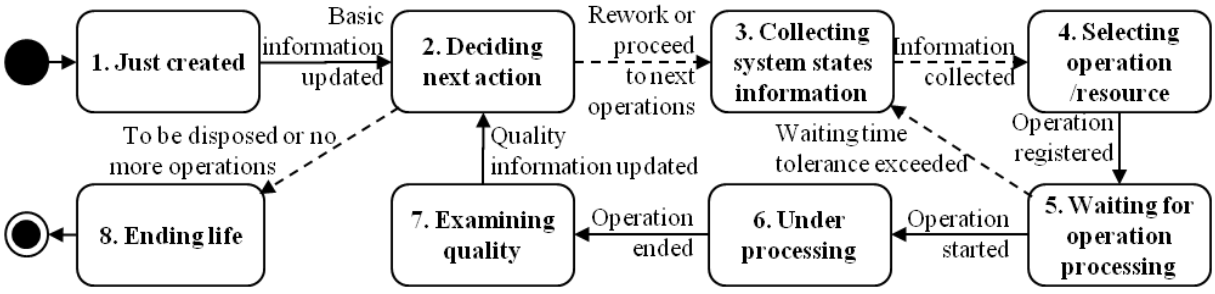


Figure V.5. State transition diagram of a PDSP agent.

[1] The line arrows (⟶) and dotted arrows (⇢) respectively indicate messages from outside of the agent and the self-generated signals.

The agent decides its own process by consecutively selecting a next operation to do after each operation completion (state 4). The resource to do the selected operation is simultaneously selected by comparing the relative expected system performance of alternative resource selection cases. Therefore the proposed multi-agent system utilizes rather a PDSP oriented scheduling mechanism than a resource oriented one which is a common characteristic of usual remanufacturing systems. The PDSP agent maintains the statistical information (refer to section IV.2) of the operation processing and the real-time information of corresponding elements in a real-world remanufacturing system: quality, current state, arrival time to a buffer, waiting time for an operation, and so on. The operation and resource are selected based on that information as the embedded decision logic. The next action to do is selected based on its current quality information among the following possible four kinds: proceed to the next operations, dispose of itself, rework the previously done operation, and end its process (refer to the state transition arrows from state 2 in figure V.5). The quality fulfilling the required criteria makes the PDSP proceed to next operations. However insufficient quality with respect to the required criteria can result in two alternative actions; the irrecoverable quality to the required level makes the PDSP be disposed of, or the previously done operation is retried in the other quality cases. Ending of its process is definitely selected in case the PDSP has no more operations to do. What a PDSP does after its creation and operation processing is the same: examine the quality and decide the next action, hence the next state of the states just created (state 1) and examining quality (state 7) is the same: the state deciding next actions (state 2).

PDSP agents are dynamically created and killed unlike the agents in other groups. The agents in the PDSP life management group create PDSP agents and start their lives in accordance with the PDSP appearances in the real-world remanufacturing system. Each PDSP agent kills itself, when it receives a disappearance signal of its corresponding real-world element. Before killing, they pass their information to the PDSP disassembly/ reassembly manager agent in the PDSP life management group, because the newly created agents for disassembled/reassembled PDSPs should inherit the information for the future decisions of resources and PDSPs.

2.1.2. Facility agent group

The facility agent group comprises the resource agents, workstation agents, and buffer agents. They correspond to the facility elements on the remanufacturing shop of the real-world remanufacturing system. These agents are created at the virtual system creation time and killed right before shutting down the virtual system.

The resource agent can have three states: 1. waiting for operation processing requests, 2. selecting PDSPs to process, and 3. under processing (refer to figure V.6). It maintains real-time information of the PDSPs in its waiting queue, which is required for applying the allocated dispatching rule. The PDSPs to-be-processed are selected based on the dispatching rule which is allocated by the belonging workstation agent. Sections VII.1 and VII.2 will explain the required PDSP information to apply dispatching rules.

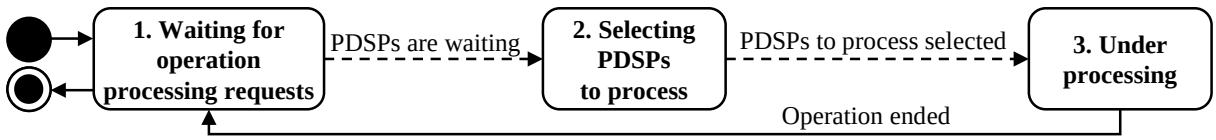


Figure V.6. State transition diagram of the resource agent.

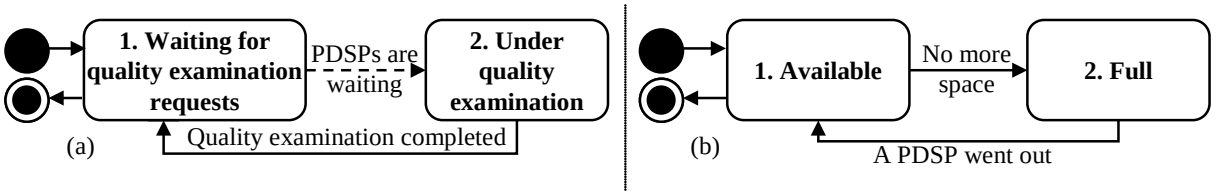


Figure V.7. State transition diagrams of (a) the workstation agent and (b) the buffer agent.

The workstation agent handles the examined PDSP quality from the real-world remanufacturing system. It requests the quality examination after operation processing, and directly passes the quality information to the PDSP agent. Hence it can have two states: 1. waiting for quality examination requests and 2. under testing (refer to figure V.7(a)). It also acts as a messenger between the dispatching rule allocator agent and the belonging resources; the dispatching rule allocator agent sends the best dispatching rule for the current system state (refer to section V.2.1.4), then the workstation agent distributes the information to belonging resources. Handing dispatching rules does not affect its state, because it is just information passing.

A buffer agent can have two states: 1. available and 2. full. Each state respectively indicates that additional PDSPs are acceptable and that no more space for PDSPs is available (refer to figure V.7(b)). The buffer agent compares the number of PDSPs in the corresponding buffer in the real-world system and its capacity whenever a PDSP comes in or goes out. Its state changes depending on the degree of capacity availability. The capacity state information is for answering to the availability checks by PDSPs wishing to come into the buffer. It maintains real-time information on the list of PDSPs waiting in the buffer.

2.1.3. PDSP life management agent group

The used product arrival manager agent and the PDSP disassembly/reassembly manager agent belong to this group. These agents are also created and killed at the same time with agents in the facility agents group.

The used product arrival manager agent creates a PDSP agent for each arrived used product in the real-world remanufacturing system. It also sets the required information for the QRS scheduling to the created PDSP agent: arrival time, due date, operations to-be-processed, and so on. The agent just waits for the used product arrivals. The arrived used products are registered to the arrival list which is managed by the product arrival manager agent, and the corresponding PDSP agents are created one by one. Hence it can have two states: 1. waiting used product arrival and 2. creating PDSP agents (refer to figure V.8(a)).

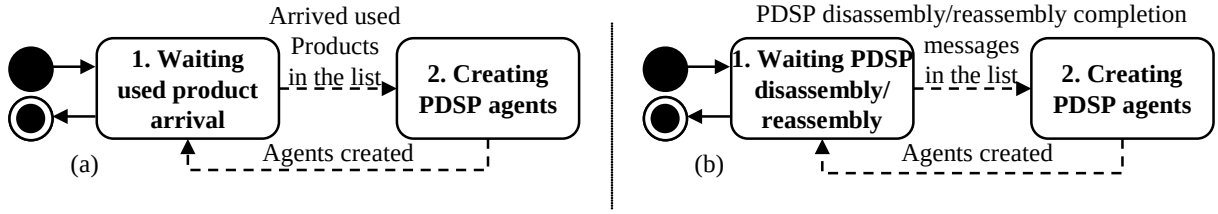


Figure V.8. State transition diagrams of (a) the used product arrival manager agent and (b) the PDSP disassembly/reassembly manager agent.

When a PDSP is disassembled into subassemblies/parts or some parts are reassembled into a product in the real-world remanufacturing system, agents corresponding to the disassembled/reassembled PDSPs should be created. The PDSP disassembly/reassembly manager agent deals with it. The created agents for the disassembled/ reassembled PDSPs should inherit some information like due date from the previous agents. This information is transferred without any transformation in the PDSP disassembly case. But the information should be consolidated in the PDSP reassembly case, because the PDSPs from different used products have different values. The aggregated information is mainly used for the parameters in the priority calculation functions of dispatching rules allocated to resources, and the reassembled PDSP's priority is preferable to be set as the average of composing PDSPs priorities. Hence this thesis simply aggregates the information by averaging the values of PDSPs to be reassembled; for example, the arrival time to the system of a reassembled product is set as the average arrival time of PDSPs to be reassembled. The received PDSP disassembly/reassembly completion message is registered to the agent's disassembly/reassembly list, and the corresponding PDSPs are created one by one in the same way with the used product arrival manager agent. Hence it can also have two states: 1. waiting PDSP disassembly/reassembly and 2. creating PDSP agents (refer to figure V.8(b)).

2.1.4. Knowledge support agent group

In this group belong the operation processing statistics information manager agent, system variable monitor agent, dispatching rule allocator agent, and simulator agent. The former three agents are also created and killed at the same time with agents in the facility agents group, but the simulator agent is created on demand of knowledge updating by the real-world QRS controller.

The operation processing statistics information manager agent keeps the statistical information on each operation processing: the processing time and POF of each operation and the resource quality (refer to section IV.2). The agent maintains the information for all combinations of operations and resources which can do the operations. The resource quality information is updated in real-time according to the operation processing results. The statistical information is transferred to the agents in the PDSP life management agent group, which transfers the information to PDSP agents at their creation time. PDSP agents utilize the information for the operation and resource selection (refer to section V.2.1.1). Operation processing results are collected in the list which the operation processing statistics information manager agent maintains, and the agent generates statistical information based on the collected operation processing results. The statistical information of the agents in the PDSP life management agent group is updated to new statistical information only when the generated statistics is different enough from the previous one. Hence it can have three states: 1. waiting operation processing results, 2. generating statistical information, and 3. updating statistical information to the agents in the PDSP life management agent group (refer to figure V.9).

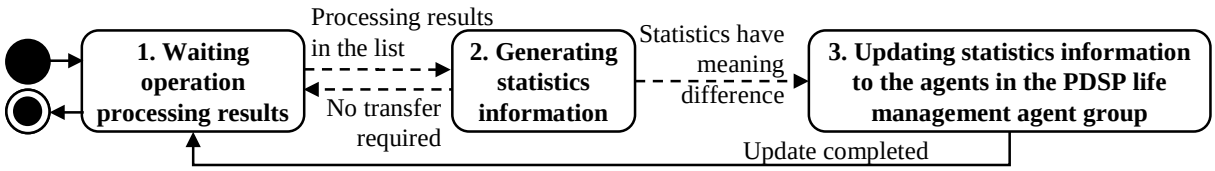


Figure V.9. State transition diagrams of the operation processing statistics information manager agent.

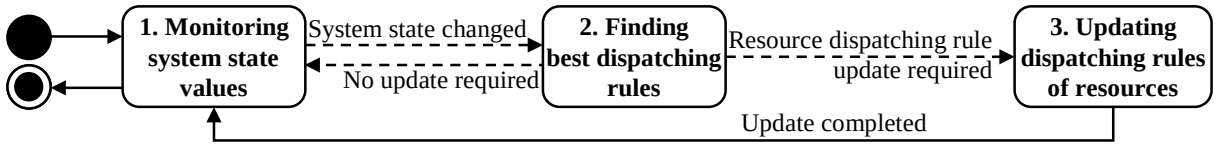


Figure V.10. State transition diagrams of the dispatching rule allocator agent.

The system variable monitor agent monitors and maintains required various raw information of the QRS in real-time; for example, the number of PDSPs before disassembly, the utilization rate of each resource, and so on. The agent gathers required information by communications with PDSP and facilities agents instead of inspecting the real-world remanufacturing system directly, because they keep real-time information synchronized with the real-world remanufacturing system. It returns the collected information upon requests from the dispatching rule allocator agent which manipulates and utilizes the gathered information for the system state values of the knowledge-based system. Section VIII.6.2 will describe the required raw information for system state variable calculations. Therefore the agent can have only one state: 1. collecting real-time information. The state transition diagram is omitted because of its simplicity.

The dispatching rule allocator agent periodically derives the best dispatching rules to be applied to resources from the perspective of the QRS performance maximization. The best dispatching rules are gathered from the accumulated knowledge by off-line simulation by the simulator agent. It also continuously monitors the system state values by communications with the system variable monitor agent, because they are raw information of input parameters for the knowledge-based decision making (refer to figures V.3 and VIII.6). The agent announces the new selected dispatching rule to the resource only in case the best dispatching rule for a resource is changed. Hence the agent can have three states: 1. monitoring system state values, 2. finding best dispatching rules, and 3. updating dispatching rules of resources (refer to figure V.10).

The simulator agent is required to generate the knowledge for the proposed scheduling mechanism. The simulator agent itself is a virtual multi-agent system which is exactly the same with the virtual system of the QRS to be simulated except for an additional module: time management module. Hence what agents do in the virtual system of the simulator agent is the same with what agents do in the real QRS except for some messages are compulsively generated by agent themselves as time goes on in the time management module; for example, the messages on the used product arrivals, operation starts and ends, and quality of a PDSP after an operation are generated by the used product arrival manager agent, resource agents, and workstation agents respectively. Such messages are generated based on the statistical information gathered from the real-world remanufacturing system executions, therefore the required statistical information should be collected before simulations. The simulator agent consolidates results from several times simulations and generates knowledge, and the generated knowledge is updated to the dispatching rule allocator agent. Hence the simulator agent can have four states: 1. collecting statistical information, 2. under simulation, 3. generating knowledge, and 4. updating knowledge to the dispatching rule allocator agent (refer to figure V.11). The simulator agent runs independently from other agents in the virtual QRS; the agent is created and starts its life when the person in charge of the QRS recognizes the necessity of knowledge updating. The IRSR model of the QRS is loaded at its creation time (refer to figure VIII.1). The agent is killed by itself after the generated knowledge is updated to the dispatching rule allocator agent.

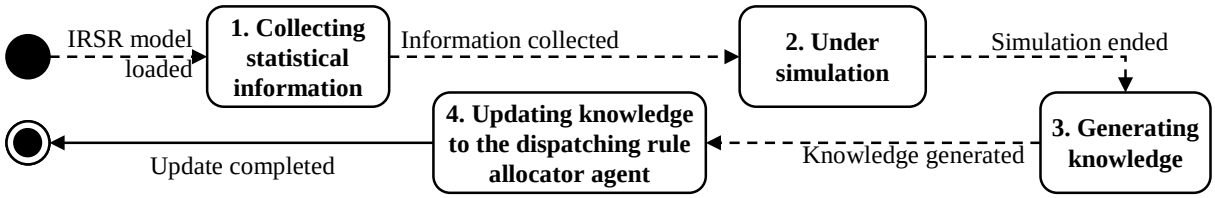


Figure V.11. State transition diagrams of the simulator agent.

2.2. Agent life cycle

The lives of all agents start and end as the virtual multi-agent system starts and ends except for PDSP agents (refer to each subsection of section V.2.1). The resource agents should be kept alive even if the corresponding resources in the real-world remanufacturing system are down, because they must respond to the other agents that they cannot serve for a moment.

The life time of most PDSP agents exactly matches with that of corresponding PDSPs in the real-world remanufacturing system except for the part reassembly cases. An agent for the reassembled product is created when the required parts are ready to be reassembled. The created agent stays as an immature agent and requests a reassembly operation to a resource. The agent starts its life when the reassembly is successfully completed. Such a way of handling the reassembled PDSP agent is for the compactness of reassembly operation requests, because it is inefficient and difficult for a resource to maintain all requests from the parts to be reassembled.

VI. Modeling Tools for the QRS

This chapter proposes two modeling tools: the intuitive remanufacturing system representation (IRSR) and the dynamic token two-level colored Petri-nets (DTPN). Their formal definitions and graphical notations are represented, and a conversion method from an IRSR model to a DTPN model is also proposed. These tools are designed for multi-agent system creation and execution. Revising the notions of states of the used products and disassembled subassemblies/parts (PDSPs) and resource agents in the multi-agent system will be helpful to follow this chapter (refer to sections V.2.1.1 and V.2.1.2).

Abbreviated terms used in this chapter:

DTPN	Dynamic Token Petri-Net;
EfS	*Exit from the System*;
EtS	*Entrance to the System*;
FCFS	First Come First Serve;
ID	IDentification;
IRSR	Intuitive Remanufacturing System Representation;
PDF	Probability Distribution Function;
PDSP	used Product and Disassembled Subassembly/Part;
POF	Probability of Operation Failure;
POS	Probability of Operation Success;
QRS	Quality embedded Remanufacturing System;
STD	STandard Deviation;
XCPN	eXtended two-level Colored Petri-Nets.

1. Fundamental modeling elements

Modeling tools for the quality embedded remanufacturing system (QRS) are developed with the elements and their attributes that are derived by decomposing a general QRS into sub-elements in a stepwise manner. They are listed in figure VI.1, where the notations in the parenthesis are corresponding elements in the intuitive remanufacturing system representation (IRSR) modeling tool which is defined in section VI.2. Some statistical elements in IRSR are defined with the PDSP/resource quality related notations discussed in chapter IV. The QRS has three constructs: a remanufacturing shop, PDSPs-alternative processes, and the combinational information which correlates the two constructs.

1.1. Remanufacturing shop: buffer, workstation, and resource

The main elements in a remanufacturing shop are buffers and workstations. Each element has an identification (ID) and a name as default information. A workstation consists of one or more resources that do identical functions, which can be humans or machines. Each resource belongs to only one workstation. Batch resources should be distinguished from other resources. Each workstation can be classified depending on its operation type in charge: disassembly, reassembly, machining, cleaning, painting, or others. Buffers have limited capacity. The capacity means the maximum number of used products or disassembled subassemblies/parts (PDSPs) containable in a buffer. Input and output buffers have to be specified for each workstation.

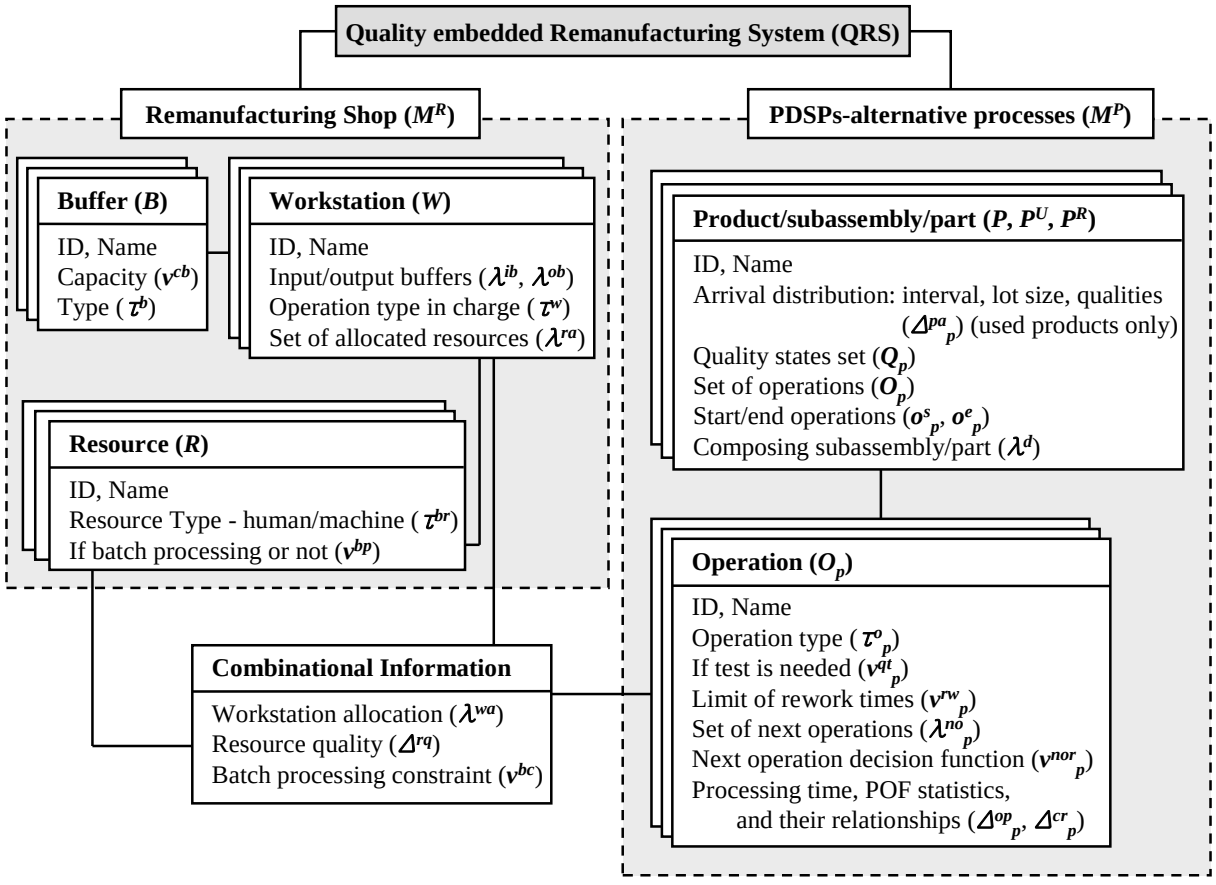


Figure VI.1. Necessary elements and attributes for the QRS representation.

1.2. Used product, subassembly/part, remanufactured product, and operation

A QRS usually contains multiple kinds of used products which are disassembled into subassemblies or parts. They have IDs and names to distinguish each other. Each PDSP has the attributes associated with the possible PDSP qualities (the quality states set in figure VI.1): the distinctive defects affect differently the operation processing time, the probability of operation failure (POF), and the output quality after the operation. The PDSP quality changes dynamically as a result of each remanufacturing operation; for instance, defects information can be changed by a physical mishandling during a disassembly operation or by uncovering hidden defects. Each PDSP has to contain the information of operations to-be-processed with the specification of start and end operations. The used products have additional statistical information on their arrivals at the QRS: arrival time interval, lot size for each arrival, and qualities distribution of arrived used products.

Each operation has information on the operation type, carrying out a PDSP quality test after the operation or not, limit number of re-work times, next possible alternative operations, and statistics of the operation processing time and the POF. The next operations of an operation mean the possible operations which can be processed right after the completion of the operation.

The QRS does not confine an operation's next operation to just one but allows alternative remanufacturing processes to be considered depending on the PDSP quality. Hence an operation should keep a set of possible next operations; for example, two alternative operations are possible after the operation OPa06 depending on the part [A2] quality of the example QRS in chapter III: the [A2] refurbishing operation OPa07 before the reassembly operation OPa08 or direct reassembly (refer to section III.2.1). [A2] having satisfactory quality to be reassembled after OPa06 does not require OPa07 and is directly sent to OPa08, but OPa07 should be carried out in the opposite case. Therefore the next alternative operations of OPa06 are OPa07 and OPa08. Two or more possible operations independent of the PDSP quality can also be handled as alternative operations; the used product [A] disassembly operations OPa01 and OPa02 of the example QRS in chapter III (refer to figure III.3) can be handled in the same way with that of OPa07 and OPa08. A next operation decision function is required to select an operation among the two or more alternative next operations. Next operation selection depending just on the PDSP quality is simple: get current quality and return the operation to-be-processed under the quality. But the operation selection with consideration of other system state factors is complicated: estimate performance of each of the candidate operations and select the operation expected to show the best performance. The performance estimation for each operation selection case will be discussed later in detail in section VIII.4.

The QRS selects the PDSP remanufacturing processes and the PDSP dispatching rules of resources based on the estimated operation processing time and the expected PDSP quality which are derived from the statistics information of the operation processing time and the POF of each operation. Hence each operation should have the statistical information on its operation processing. Even though the processing statistic is different depending on not only the PDSP characteristics but also the characteristics of the resources which are in charge of operations, this thesis divides the information into two parts as discussed in section IV.2; an operation itself maintains the statistics in the average case, and the effect by resource quality is used as additional information. Therefore operations in the PDSPs-alternative processes side only keep the average operation processing statistics.

1.3. Combinational information

Some information can be defined only by a combination of elements in a remanufacturing shop and PDSPs-alternative processes: workstations in charge of operations, resource quality for operations, and batch processing conditions. A PDSP can select an operation and a resource to do the operation with the information on the workstations allocation to operations and the resource quality for operations. Each batch resource requires information on the processing condition depending on operations or operation combinations: the possibility of simultaneous multiple PDSPs handling, the capacity calculation method, and so on. The batch processing conditions vary from resource to resource, hence this thesis does not suggest a general format but just leave it as a free function. An example batch processing condition function is represented in appendix E.3 for the batch resources in the Example QRS in chapter III.

2. IRSR: Intuitive Remanufacturing System Representation

2.1. Formal definition of IRSR

Some type of information should be specified to the elements in the IRSR model. Here we define the sets of element types which are involved in the IRSR definition as follows:

$\boldsymbol{T^R}$ = {*human*, *machine*}; (Def. VI.2.1)

$\boldsymbol{T^W}$ = {*disassembly*, *refurbishing*, *cleaning*, *painting*, *reassembly*, *testing*, *packaging*}; (Def. VI.2.2)

$\boldsymbol{T^B}$ = {*entrance to the system* (*EtS*), *exit from the system* (*EfS*), *inside*}; (Def. VI.2.3)

$\boldsymbol{T^O}$ = {*disassembly*, *substitution*, *refurbishing*, *cleaning*, *painting*, *reassembly*, *testing*, *packaging*, *virtual*}. (Def. VI.2.4)

$\boldsymbol{T^R}$, $\boldsymbol{T^W}$, $\boldsymbol{T^B}$, and $\boldsymbol{T^O}$ are respectively physical types of resources, logical types of workstations, buffer types, and operation types. Here the *substitution* type in $\boldsymbol{T^O}$ means a disassembled PDSP is substituted with a new one by the operation. This type is applicable to the PDSP which needs to be replaced with a new PDSP right after the disassembly completion of its parent PDSP for the reassembly without quality examination.

The *virtual* type in $\boldsymbol{T^O}$ is required to handle the alternative operations right after PDSP disassembly. For example, modeling of remanufacturing process after the disassembly operation OPa03 of the example QRS in chapter III is complicated and ambiguous (refer to figure VI.2(a)), because the next operations of disassembled [A1] and [A2] are different depending on their quality. The representation without *virtual* operation can be misunderstood as that two [A1] and three [A2] come out from the [A12][1] disassembly operation, while no other interpretation is possible for the modeling with *virtual* operation (refer to figure VI.2(b)). Hence this thesis adopts an intermediate *virtual* type operation for the simple and obvious representation.

In the following IRSR definition, the notations $\boldsymbol{\lambda}$, $\boldsymbol{\Delta}$, $\boldsymbol{\nu}$, and $\boldsymbol{\tau}$ are respectively used for the specification of relationships among modeling elements, the statistical information, the value or function, and the element type.

[1] [A12] means the subassembly composed of [A1] and [A2]. This subassembly representation method is valid for the whole document.

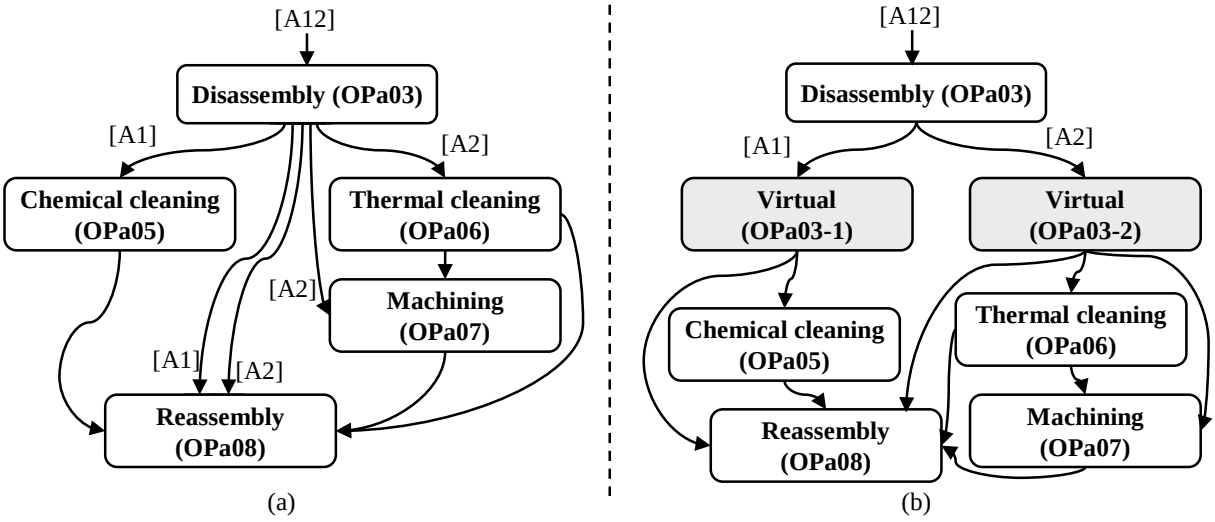


Figure VI.2. The necessity of the virtual operation; (a) disassembly operation representation without a virtual operation; (b) disassembly operation representation with virtual operations.

2.1.1. Intuitive QRS representation model: IRSR

The ***IRSR*** model is defined as a 5-tuple:

$$\boldsymbol{IRSR} = \{\boldsymbol{M^R}, \boldsymbol{M^P}, \boldsymbol{\lambda^{wa}}, \boldsymbol{\Delta^{rq}}, \boldsymbol{v^{bc}}\}, \quad \text{(Def. VI.2.5)}$$

where

$\boldsymbol{M^R}$ remanufacturing shop model which represents facilities and their relationships; (Def. VI.2.6)

$\boldsymbol{M^P}$ PDSPs-alternative processes model which represents products and their detailed operations. (Def. VI.2.7)

$\boldsymbol{\lambda^{wa}}$ and $\boldsymbol{\Delta^{rq}}$ are for the workstations allocation to operations and the resource quality specification to the combinations of operations and resources, respectively. $\boldsymbol{v^{bc}}$ is required to define batch processing constraints. The three elements will be defined later in detail, because the definition requires elements of $\boldsymbol{M^R}$ and $\boldsymbol{M^P}$.

2.1.2. Remanufacturing shop model - $\boldsymbol{M^R}$

$\boldsymbol{M^R}$ is defined as an 11-tuple:

$$\boldsymbol{M^R} = \{\boldsymbol{W}, \boldsymbol{R}, \boldsymbol{B}, \boldsymbol{\lambda^{ra}}, \boldsymbol{\lambda^{ib}}, \boldsymbol{\lambda^{ob}}, \boldsymbol{v^{bp}}, \boldsymbol{v^{cb}}, \boldsymbol{\tau^{w}}, \boldsymbol{\tau^{br}}, \boldsymbol{\tau^{b}}\}.$$

The first three elements are sets containing the following elements:

$\boldsymbol{W}$	workstations;	(Def. VI.2.8)
$\boldsymbol{R}$	resources;	(Def. VI.2.9)
$\boldsymbol{B}$	buffers,	(Def. VI.2.10)

and the other elements are functions which specify the following attributes:

λ^{ra}: $\boldsymbol{R} \rightarrow \boldsymbol{W}$	workstation to which resources belong;	(Def. VI.2.11)
λ^{ib}: $\boldsymbol{W} \rightarrow 2^{\boldsymbol{B}}$	input buffers of a workstation;	(Def. VI.2.12)
λ^{ob}: $\boldsymbol{W} \rightarrow 2^{\boldsymbol{B}}$	output buffers of a workstation;	(Def. VI.2.13)
ν^{bp}: $\boldsymbol{W} \rightarrow \mathbb{B}$	if a workstation contains batch resources or not;	(Def. VI.2.14)
ν^{cb}: $\boldsymbol{B} \rightarrow \mathbb{I}^{+} \cup \{\boldsymbol{inf.}\}$	capacity of each buffer;	(Def. VI.2.15)
τ^{w}: $\boldsymbol{W} \rightarrow \boldsymbol{T}^{\boldsymbol{W}}$	functional type of operations which a workstation is in charge of;	(Def. VI.2.16)
τ^{br}: $\boldsymbol{W} \rightarrow \boldsymbol{T}^{\boldsymbol{R}}$	physical type of belonging resources;	(Def. VI.2.17)
τ^{b}: $\boldsymbol{B} \rightarrow \boldsymbol{T}^{\boldsymbol{B}}$	type of each buffer.	(Def. VI.2.18)

The type of all resources in a workstation is the same (refer to section VI.1.1), therefore the source domain of the resource type specification function τ^{br} is not the set of resources $\boldsymbol{R}$ but the set of workstations $\boldsymbol{W}$ for the model conciseness and consistency. Hence the type of resource $\boldsymbol{r}$ can be gathered by application of two functions τ^{br} and the belonging resource specification function λ^{ra} as $\tau^{br}(\lambda^{ra}(\boldsymbol{r}))$. The source domain of the batch resource indication ν^{bp} is also $\boldsymbol{W}$ for the same reason.

The sound remanufacturing shop model should keep some constraints. All elements in the QRS must exclusively belong to one of the three sets $\boldsymbol{W}$, $\boldsymbol{R}$ and $\boldsymbol{B}$ (constraint VI.2.1), which are not empty (constraint VI.2.2). All elements in $\boldsymbol{W}$, $\boldsymbol{R}$, and $\boldsymbol{B}$ should have relationships with other elements; all resources should belong to workstations, all workstations should have input/output buffers (constraint VI.2.3), and all buffers should be linked to at least one workstation (constraint VI.2.4). While each workstation's input and output buffers are one or more, a resource can belong to only one workstation (constraint VI.2.5). The QRS should have connections to the outside environment: the entrance and exit buffers, types of which are $EtS \in \boldsymbol{T}^{\boldsymbol{B}}$ (def. VI.2.3) and $EfS \in \boldsymbol{T}^{\boldsymbol{B}}$ respectively (constraint VI.2.6 and VI.2.7), and those buffers cannot be specified respectively as output buffers or input buffers of workstations (constraint VI.2.8 and VI.2.9). But an *inside* buffer should be linked with both at least one workstation as an input buffer and at least one workstation as an output buffer (constraint VI.2.10). The above discussed constraints are defined as follows:

- $\boldsymbol{S} \neq \varnothing$, $\boldsymbol{R} \neq \varnothing$, and $\boldsymbol{B} \neq \varnothing$; (Constraint VI.2.1)
- $\boldsymbol{S} \cap \boldsymbol{R} \cap \boldsymbol{B} = \varnothing$; (Constraint VI.2.2)
- $\forall \boldsymbol{w} \in \boldsymbol{W}$: $\lambda^{ib}(\boldsymbol{w}) \neq \varnothing$ and $\lambda^{ob}(\boldsymbol{w}) \neq \varnothing$; (Constraint VI.2.3)
- $\forall \boldsymbol{b} \in \boldsymbol{B}$: $\boldsymbol{b} \in \lambda^{ib}(\exists \boldsymbol{w}_i \in \boldsymbol{W})$ or $\boldsymbol{b} \in \lambda^{ob}(\exists \boldsymbol{w}_j \in \boldsymbol{W})$; (Constraint VI.2.4)
- $\forall \boldsymbol{r} \in \boldsymbol{R}$: $|\lambda^{ra}(\boldsymbol{r})| = 1$; (Constraint VI.2.5)
- $\exists \boldsymbol{b} \in \boldsymbol{B}$: $\tau^{b}(\boldsymbol{b}) = EtS$; (Constraint VI.2.6)
- $\exists \boldsymbol{b} \in \boldsymbol{B}$: $\tau^{b}(\boldsymbol{b}) = EfS$; (Constraint VI.2.7)

– $\forall \boldsymbol{b}\in\boldsymbol{B}$: $\boldsymbol{\tau}^{b}(\boldsymbol{b}) = EtS \Leftrightarrow \boldsymbol{b}\in\boldsymbol{\lambda}^{ib}(\exists \boldsymbol{w}_i\in\boldsymbol{W})$ and $\boldsymbol{b}\in\boldsymbol{\lambda}^{ob}(\sim\exists \boldsymbol{w}_j\in\boldsymbol{W})$; (Constraint VI.2.8)

– $\forall \boldsymbol{b}\in\boldsymbol{B}$: $\boldsymbol{\tau}^{b}(\boldsymbol{b}) = EfS \Leftrightarrow \boldsymbol{b}\in\boldsymbol{\lambda}^{ib}(\sim\exists \boldsymbol{w}_i\in\boldsymbol{W})$ and $\boldsymbol{b}\in\boldsymbol{\lambda}^{ob}(\exists \boldsymbol{w}_j\in\boldsymbol{W})$; (Constraint VI.2.9)

– $\forall \boldsymbol{b}\in\boldsymbol{B}$: $\boldsymbol{\tau}^{b}(\boldsymbol{b}) = inside \Leftrightarrow \boldsymbol{b}\in\boldsymbol{\lambda}^{ib}(\exists \boldsymbol{w}_i\in\boldsymbol{W})$ and $\boldsymbol{b}\in\boldsymbol{\lambda}^{ob}(\exists \boldsymbol{w}_j\in\boldsymbol{W})$. (Constraint VI.2.10)

2.1.3. PDSPs-alternative processes model - $\boldsymbol{M}^{\boldsymbol{P}}$

$\boldsymbol{M}^{\boldsymbol{P}}$ is defined as a 5-tuple:

$\boldsymbol{M}^{\boldsymbol{P}} = \{\boldsymbol{P}, \boldsymbol{P}^{U}, \boldsymbol{P}^{R}, \boldsymbol{\lambda}^{d}, \boldsymbol{\lambda}^{cq}\}$,

and each element $\boldsymbol{p}\in\boldsymbol{P}$ is defined as a 12-tuple:

$\boldsymbol{p} = \{\boldsymbol{Q}_p, \boldsymbol{O}_p, \boldsymbol{o}^{s}_{p}, \boldsymbol{o}^{e}_{p}, \boldsymbol{\lambda}^{no}_{p}, \boldsymbol{\Delta}^{pa}_{p}, \boldsymbol{\Delta}^{op}_{p}, \boldsymbol{\Delta}^{cr}_{p}, \boldsymbol{v}^{qt}_{p}, \boldsymbol{v}^{rw}_{p}, \boldsymbol{v}^{nor}_{p}, \boldsymbol{\tau}^{o}_{p}\}$, (Def. VI.2.19)

where

$\boldsymbol{P}$	set of PDSPs in a QRS;	(Def. VI.2.20)
$\boldsymbol{P}^{U}\subset\boldsymbol{P}$	set of used products;	(Def. VI.2.21)
$\boldsymbol{P}^{R}\subset\boldsymbol{P}$	set of remanufactured products;	(Def. VI.2.22)
$\boldsymbol{\lambda}^{d}: \cup\boldsymbol{O}_p \rightarrow 2^{\boldsymbol{P}}$	function specifying disassembled subassemblies/parts or reassembled products through a disassembly/reassembly operation;	(Def. VI.2.23)
$\boldsymbol{\lambda}^{cq}: \cup\boldsymbol{Q}_p\times\boldsymbol{P} \rightarrow \cup\boldsymbol{Q}_p$	function specifying the corresponding qualities of $\boldsymbol{p}$'s composing PDSPs to $\boldsymbol{p}$'s quality;	(Def. VI.2.24)
$\boldsymbol{Q}_p$	possible PDSP quality set (def. IV.2.1);	
$\boldsymbol{O}_p$	set of operations in all alternative processes;	(Def. VI.2.25)
$\boldsymbol{o}^{s}_{p}\in 2^{Op}$	start operations of $\boldsymbol{p}$;	(Def. VI.2.26)
$\boldsymbol{o}^{e}_{p}\in 2^{Op}$	end operations of $\boldsymbol{p}$;	(Def. VI.2.27)
$\boldsymbol{\lambda}^{no}_{p}: \boldsymbol{O}_p \rightarrow 2^{Op}$	function specifying possible next operations of each operation;	(Def. VI.2.28)
$\boldsymbol{\Delta}^{pa}_{p} = (\boldsymbol{ai}_p, \boldsymbol{ls}_p, \boldsymbol{qd}_p)$	statistical information of used product arrival;	(Def. VI.2.29)
$\boldsymbol{\Delta}^{op}_{p} = \{(\boldsymbol{o}, \boldsymbol{iq}, \boldsymbol{tf}_{o,iq}, \boldsymbol{RPQ}^{nc,d}_{o,iq}, \boldsymbol{RPQ}^{c,d}_{o,iq}, \boldsymbol{RPQ}^{c}_{o,iq}) \mid \boldsymbol{o}\in\boldsymbol{O}_p$ and $\boldsymbol{iq}\in\boldsymbol{Q}_p\}$	set of statistical information of the operation processing time and POF $\boldsymbol{tf}_{o,iq}$ (def. IV.2.2) and the quality distribution after the operation $\boldsymbol{RPQ}^{*}_{o,iq}$ (def. IV.2.9 and def. IV.2.11 – IV.2.13) of an operation $\boldsymbol{o}$ with input PDSP quality $\boldsymbol{iq}$;	(Def. VI.2.30)
$\boldsymbol{\Delta}^{cr}_{p} = \{(\boldsymbol{o}, \boldsymbol{CR}_o) \mid \boldsymbol{o}\in\boldsymbol{O}_p\}$	set of statistical information of the correlation $\boldsymbol{CR}_o$ (def. IV.2.14) of an operation $\boldsymbol{o}$ between the operation processing time and output PDSP quality;	(Def. VI.2.31)
$\boldsymbol{v}^{qt}_{p}: \boldsymbol{O}_p \rightarrow \mathbb{B}$	function specifying the PDSP quality examination necessity after an operation;	(Def. VI.2.32)
$\boldsymbol{v}^{rw}_{p}: \boldsymbol{O}_p \rightarrow \mathrm{I}^{0+}$	function specifying the limit number of rework times of an operation;	(Def. VI.2.33)

$\boldsymbol{v}^{nor}_{p} = \{(\boldsymbol{o}, \boldsymbol{f}^{nor}_{o}) \mid \boldsymbol{o} \in \boldsymbol{O}_p$ and $\boldsymbol{f}^{nor}_{o}: \boldsymbol{p} \times 2^{\mathbb{R}} \rightarrow \boldsymbol{O}_p \times \boldsymbol{R}\}$

set of selection functions of each PDSP for the next operation of an operation $\boldsymbol{o}$ and resource to do the selected operation; (Def. VI.2.34)

$\boldsymbol{\tau}^{o}_{p}: \boldsymbol{O}_p \rightarrow \boldsymbol{T}^{O}$ function specifying the type of each operation, (Def. VI.2.35)

where

$\boldsymbol{ai}_p \in \mathbf{F}^{\mathbf{PDF}}$ statistical information of the arrival interval of used products; (Def. VI.2.36)

$\boldsymbol{ls}_p \in \mathbf{F}^{\mathbf{PDF}}$ statistical information of the lot size of used products at each arrival time; (Def. VI.2.37)

$\boldsymbol{qd}_p = \{(\boldsymbol{aq}, \boldsymbol{rto}_{aq}) \mid \boldsymbol{aq} \in \boldsymbol{Q}_p, 0 < \boldsymbol{rto}_{aq} \leq 1$, and $\boldsymbol{\Sigma}_{aq}\boldsymbol{rto}_{aq} = 1\}$

statistical information of the arrived used product quality frequency ratio; (Def. VI.2.38)

$\boldsymbol{f}^{nor}_{o}: \boldsymbol{Q}_p \times 2^{\mathbb{R}} \rightarrow \boldsymbol{O}_p \times \boldsymbol{R}$ function selecting a next operation to-be-processed and a resource to handle the selected operation based on not only the PDSP quality but also the set of real numbers which express the QRS real-time states and the historical resource quality, (Def. VI.2.39)

where

$\boldsymbol{aq}$ used product quality at arrival time;

$\boldsymbol{rto}_{aq} \in \mathbb{R}^{\mathbf{0+}}$ occurrence frequency of used product quality $\boldsymbol{aq}$ at arrival time. (Def. VI.2.40)

The quality mapping function $\boldsymbol{\lambda}^{\boldsymbol{cq}}$ is required to handle the quality of disassembled PDSPs as explained in section IV.2.1. Figure VI.3 represents a usage example of $\boldsymbol{\lambda}^{\boldsymbol{cq}}$ with the disassembly of [A12] in the example QRS in chapter III; a subassembly [A12] having a defect of its composing two parts' adherence (*J1AH*) is separated into a [A1] and a [A2] by the operation OPa03, and the disassembly caused the crack and surface scratch of the disassembled [A1] and [A2] respectively. The quality of the disassembled [A1] and [A2] right after disassembly are integrated and indicated together as one value: *A1CoW_A2SoG*, and the quality mapping function $\boldsymbol{\lambda}^{\boldsymbol{cq}}$ is applied to get the quality of each disassembly part: $CoW = \boldsymbol{\lambda}^{\boldsymbol{cq}}(A1CoW\_A2SoG, [A1])$ for the [A1] quality and $SoG = \boldsymbol{\lambda}^{\boldsymbol{cq}}(A1CoW\_A2SoG, [A2])$ for the [A2] quality, which means the crack of [A1] and the surface scratch of [A2] respectively. The return value of $\boldsymbol{\lambda}^{\boldsymbol{cq}}$ definitely should be in the possible quality set of each disassembled parts; $CoW \in \boldsymbol{Q}_{[A1]}$ and $SoG \in \boldsymbol{Q}_{[A2]}$.

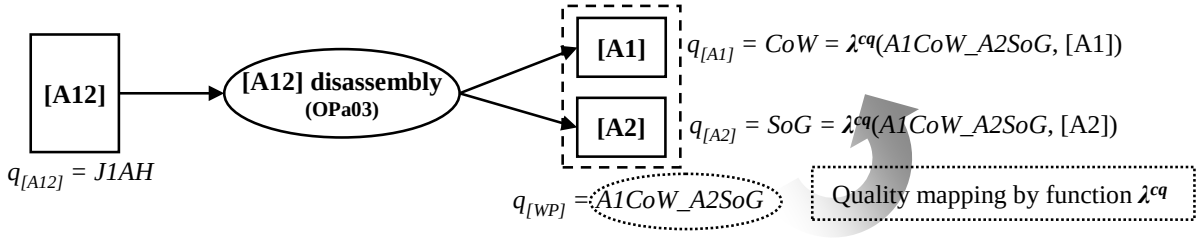


Figure VI.3. Quality of disassembled PDSPs after a disassembly operation.

The alternative remanufacturing processes can be formalized and handled by the next operation specification function λ^{no}_{p} and the next operation/resource selection function f^{nor}_{o}. Possible operations after completion of an operation are gathered by λ^{no}_{p}, and an operation and a resource to do the operation are selected by f^{nor}_{o}. This research leaves f^{nor}_{o} as a free form which can utilize any information in the QRS like the PDSP quality, number of PDSPs in the resource's waiting queue, and so on, because the decision making algorithm can be too complicated to be confined in a regular simple formula. Here a simple example is presented. The decision function f^{nor}_{OPa06} of the thermal cleaning operation OPa06 of the example QRS in chapter III can be defined as follows:[1]

$$f^{nor}_{OPa06}(oq, n^{trial}, r) = \begin{pmatrix} (\text{OPa06, RC5\_1}) & \textbf{, if } oq = UIQ, n^{trial} \le 2, \textbf{ and } r = \text{RC5\_2} \\ (\text{OPa06, RC5\_2}) & \textbf{, elseif } oq = UIQ, n^{trial} \le 2, \textbf{ and } r = \text{RC5\_1} \\ (\text{OPa07, RC4\_1}) & \textbf{, elseif } oq = SoG \\ (\text{OPa08, RC7\_1}) & \textbf{, elseif } oq = ND \\ \textbf{null} & \textbf{, elsewise} \end{pmatrix},$$

where

oq output quality of the operation in the previous trial;

n^{trial} next number of operation trial;

r resource which did the operation in the previous trial.

The first and second conditions are for the unsuccessful cleaning cases. The return values of the decision function f^{nor}_{OPa06} mean that the operation should be redone by the resource which did not handle the previous trial. The third and fourth conditions are the cases of successful cleaning without critical defects; the part [A2] having a surface scratch or a weak wear of the grooves should undergo the refurbishing operation OPa07, while [A2] having no defects can be directly reassembled by the reassembly operation OPa08. The last condition is the case the PDSP should be disposed of. Note that in this example λ^{no}_{p}(OPa06) = {OPa07, OPa08}, because rework is considered as an internal operation in this thesis.

A sound PDSPs-alternative processes model should keep some constraints. Any PDSPs in the remanufacturing system has at least one quality instance even though the quality is not examined (constraint VI.2.11). The possible quality set Q_p of the PDSP without quality examination can be defined as only having *UID* which means the unidentified quality as an example. For every used product handled in the remanufacturing system there should be a corresponding remanufactured product in the system. Hence the cardinalities of the used product set P^U and the remanufactured product set P^R are the same (constraint VI.2.12). Used products must be distinguished from subassemblies and parts for their special characteristics; stochastic arrival information into the QRS (constraint VI.2.13) and no parents in the product structure (constraint VI.2.14). The remanufactured product cannot include any disassembly or reassembly operation (constraint VI.2.15). This thesis treats used products to be remanufactured and reassembled products as different PDSPs when composing the PDSP set P,

[1] The possible quality set of [A2] is $Q_{[A2]}$ = {*UIQ, ND, CoG, SoG*}, each element of which means unidentified, no defects, crack or strong wear of the grooves, and surface scratch or weak wear of the grooves (refer to section IV.2.1). The operations OPa06, OPa07, and OPa08 can be respectively done by the workstations WS5, WS4, and WS7 (refer to table III.4 and figure III.1).

and disassembled subassemblies and the reassembled subassemblies are also treaded as different PDSPs. Despite their identical structure, it is reasonable to consider that a reassembled product is newly produced and a used product disappears by the disassembly operations. Therefore the disassembly/reassembly of a PDSP means the end of its life (constraint VI.2.16), and the next operation of a PDSP's end operation $\boldsymbol{o}^{e}_{p}$ should be ***null*** (constraint VI.2.17). The proposed model defines the PDSP relationship function $\boldsymbol{\lambda}^{d}$ (refer to product/subassembly/part part in figure VI.1). $\boldsymbol{\lambda}^{d}$ makes it possible to represent the alternative disassembly/ reassembly sequences as well as the product disassembly/reassembly structures. Therefore the function $\boldsymbol{\lambda}^{d}$ is valid only for the disassembly/reassembly operations and returns ***null*** for other operations (constraint VI.2.18). In addition, a PDSP itself cannot be the disassembled or assembled PDSP of the PDSP itself (constraint VI.2.19). IRSR considers operation reworks as an internal characteristic of the operation. Therefore an operation cannot be included in a set of next operations of itself (constraint VI.2.20). A rework is possible only after quality examination (constraint VI.2.21). The next operation/resource selection function $\boldsymbol{f}^{nor}_{o}$ should be definitely specified to the operation which has alternative next operations or two or more resources can process the next operations (constraint VI.2.22). The above discussed constraints are defined as follows:

- $\boldsymbol{Q}_{p} \neq \varnothing$; (Constraint VI.2.11)
- $|\boldsymbol{P}^{U}| = |\boldsymbol{P}^{R}|$; (Constraint VI.2.12)
- $\forall \boldsymbol{p} \notin \boldsymbol{P}^{U}: \boldsymbol{\Delta}^{pa}_{p} = \boldsymbol{null}$; (Constraint VI.2.13)
- $\forall \boldsymbol{up} \in \boldsymbol{P}^{U}: \boldsymbol{up} \notin \boldsymbol{\lambda}^{d}(\sim\exists \boldsymbol{p} \in \boldsymbol{P})$; (Constraint VI.2.14)
- $\forall \boldsymbol{p} \in \boldsymbol{P}^{R}: \boldsymbol{\tau}^{o}_{p}(\sim\exists \boldsymbol{o} \in \boldsymbol{O}_{p}) \in \{disassembly, reassembly\}$; (Constraint VI.2.15)
- $\boldsymbol{\tau}^{o}_{p}(\forall \boldsymbol{o} \in \boldsymbol{O}_{p}) \in \{disassembly, reassembly\} \Rightarrow \boldsymbol{o} \in \boldsymbol{o}^{e}_{p}$; (Constraint VI.2.16)
- $\boldsymbol{\lambda}^{no}_{p}(\boldsymbol{o}^{e}_{p}) = \varnothing$; (Constraint VI.2.17)
- $\forall \boldsymbol{o} \in \boldsymbol{O}_{p}: \boldsymbol{\tau}^{o}_{p}(\boldsymbol{o}) \notin \{disassembly, reassembly\} \Rightarrow \boldsymbol{\lambda}^{d}(\boldsymbol{o}) = \varnothing$; (Constraint VI.2.18)
- $\forall \boldsymbol{p} \in \boldsymbol{P}: \boldsymbol{p} \notin \boldsymbol{\lambda}^{d}(\forall \boldsymbol{o} \in \boldsymbol{O}_{p})$; (Constraint VI.2.19)
- $\forall \boldsymbol{o} \in \boldsymbol{O}_{p}: \boldsymbol{o} \notin \boldsymbol{\lambda}^{no}_{p}(\boldsymbol{o})$; (Constraint VI.2.20)
- $\forall \boldsymbol{o} \in \boldsymbol{O}_{p}: \boldsymbol{v}^{qt}_{p}(\boldsymbol{o}) = false \Rightarrow \boldsymbol{v}^{rw}_{p}(\boldsymbol{o}) = 0$; (Constraint VI.2.21)
- $\forall \boldsymbol{o} \in \boldsymbol{O}_{p}: |\boldsymbol{\lambda}^{no}_{p}(\boldsymbol{o})| > 1$ or $|\boldsymbol{\lambda}^{ra\text{-}1}(\boldsymbol{w} \in \boldsymbol{\lambda}^{wa}(\exists \boldsymbol{o}^{n}_{i} \in \boldsymbol{\lambda}^{no}_{p}(\boldsymbol{o})))| \Rightarrow (\boldsymbol{o}, \boldsymbol{f}^{nor}_{o}) \in \boldsymbol{v}^{nor}_{p}$, (Constraint VI.2.22)

where

$\boldsymbol{o}^{n}_{i}$ possible next operation of an operation $\boldsymbol{o}$.

2.1.4. Combinational elements between $\boldsymbol{M}^{R}$ and $\boldsymbol{M}^{P}$

The combinational elements of ***IRSR*** which link $\boldsymbol{M}^{R}$ and $\boldsymbol{M}^{P}$ are defined as follows:

$\boldsymbol{\lambda}^{wa}: \cup \boldsymbol{O}_{p} \rightarrow 2^{W}$ function for the workstations allocation to each operation; (Def. VI.2.41)

$\boldsymbol{\Delta}^{rq} = \{(\boldsymbol{o}, \boldsymbol{r}, \boldsymbol{rq}_{o,r}) \mid \boldsymbol{o} \in \cup \boldsymbol{O}_{p}$, and $\boldsymbol{r} \in \boldsymbol{R}\}$

set of resource quality $\boldsymbol{rq}_{o,r}$ (def. IV.2.16) for the operation processing time and the POF to operations of a resource $\boldsymbol{r}$ for an operation $\boldsymbol{o}$; (Def. VI.2.42)

$\boldsymbol{v}^{bc}$: {($\boldsymbol{r}$, $\boldsymbol{O}^{B}$, $\boldsymbol{f}^{bc}_{r,O^{B}}$) | $\boldsymbol{r}\in\boldsymbol{R}$ and $\boldsymbol{O}^{B}\subset 2^{\cup Op}$}

set of batch capacity calculation functions of the combination of $\boldsymbol{r}$ and a operation set, (Def. VI.2.43)

where

$\boldsymbol{O}^{B}$ set of operations which can be simultaneously processed by a batch resource; (Def. VI.2.44)

$\boldsymbol{f}^{bc}_{r,O^{B}}$ batch capacity calculation function of a resource $\boldsymbol{r}$ for the operation set $\boldsymbol{O}^{B}$. (Def. VI.2.45)

The resource $\boldsymbol{r}$ in an instance of the resource quality set $\boldsymbol{\Delta}^{rq}$ should be able to process the operation $\boldsymbol{o}$ in the same instance, hence the operation and resource in an instance should be related by the workstation-operation relationship function $\boldsymbol{\lambda}^{wa}$ and the resource-workstation relationship function $\boldsymbol{\lambda}^{ra}$ (def. VI.2.11) (constraint VI.2.23). The batch processing conditions should be applied only to batch resources, and all batch resources should have batch processing conditions (constraint VI.2.24). A next operation of an operation in a used PDSPs-alternative processes model $\boldsymbol{M}^{P}$ can be specified only when one of output buffers of workstations in charge of the operation is an input buffer of workstations in charge of the next operation in the remanufacturing shop model $\boldsymbol{M}^{R}$ (constraint VI.2.25). Last, the resource $\boldsymbol{r}$ in an instance of the batch capacity calculation function set $\boldsymbol{v}^{bc}$ should be able to handle the operations in the simultaneously process-able operation set $\boldsymbol{O}^{B}$ in the same instance (constraint VI.2.26). The discussed constraints are defined as follows:

– $\forall(\boldsymbol{o}, \boldsymbol{r}, \boldsymbol{rp}_{o,r})\in\boldsymbol{\Delta}^{rq}: \boldsymbol{\lambda}^{ra}(\boldsymbol{r})\in\boldsymbol{\lambda}^{wa}(\boldsymbol{o})$; (Constraint VI.2.23)

– $\forall(\boldsymbol{r}, \boldsymbol{O}^{B})\in\boldsymbol{v}^{bc} \Leftrightarrow \boldsymbol{v}^{bp}(\boldsymbol{\lambda}^{wa}(\boldsymbol{r})) = true$; (Constraint VI.2.24)

– $\forall\boldsymbol{o}_i,\boldsymbol{o}_j\in\boldsymbol{O}_p, \boldsymbol{i}\neq\boldsymbol{j}: \boldsymbol{o}_j\in\boldsymbol{\lambda}^{no}{}_{p}(\boldsymbol{o}_i) \Rightarrow \boldsymbol{\lambda}^{ob}(\boldsymbol{\lambda}^{wa}(\boldsymbol{o}_i))\cap\boldsymbol{\lambda}^{ib}(\boldsymbol{\lambda}^{wa}(\boldsymbol{o}_j)) \neq \varnothing$; (Constraint VI.2.25)

– $\forall(\boldsymbol{r}, \boldsymbol{O}^{B})\in\boldsymbol{v}^{bc}: \boldsymbol{\lambda}^{ra}(\boldsymbol{r})\in\boldsymbol{\lambda}^{wa}(\forall\boldsymbol{o}\in\boldsymbol{O}^{B})$. (Constraint VI.2.26)

2.2. Graphical notations of IRSR

Modeling a target system in the formal definition form is time consuming and arduous, hence IRSR supports graphical notations to overcome the inconvenience. The graphical notations include most features of the formal definition except for complex mathematical forms or too much information to represent; for instance, the capacity calculation functions for batch resources, the correlation information between the operation processing time and the output PDSP quality, and the performance of resources. The graphical notations should be as simple as possible to increase understandability by not only humans but also computer systems for the subsequent information processing.

2.2.1. Graphical notation of $\boldsymbol{M}^{R}$

The resource model $\boldsymbol{M}^{R}$ has three graphical notations: workstations, buffers, and relationships between buffers and workstations (refer to figure VI.4(a)). A workstation is represented by a box with a horizontal line. Batch

workstations can be distinguished by the rounded B symbol marked below the box. The place for the symbol is empty in case of the non-batch workstations. The belonging resources of a workstation are described to the right of the rounded R symbol marked right beside the batch processing symbol. A rounded box is the graphical notation for a buffer of which the capacity ***c*** is marked below the box. The ID and name of workstations and buffers are expressed in the box with icons which represent the types of workstations and buffers. Input and output buffers are linked with arrows where transferring PDSPs are marked.

2.2.2. Graphical notation of M^P

The PDSPs-alternative processes model M^P is a set of remanufacturing process models of used products. It contains the used products' whole alternative processes in the QRS. Remanufacturing process models are composed of operations and their relationships, hence M^P has two graphical notations: operations and relationships between them (refer to figure VI.4(b)). A box with a type icon and ID/name stands for an operation, and arrows specify next operations. Carrying out the quality examination after the operation, the limit number of rework times ***rw***, the probability of operation success (POS) ***pos***, and the probability distribution function (PDF) of operation processing time are marked below the box with symbols of a rounded T, a rounded W, a rounded S, and a clock shape one respectively. The average values of required PDF parameters over all possible input/output PDSP quality combinations are marked for the processing time distribution. Disassembled subassemblies/parts or a reassembled product are marked on the arrows specifying next operations of disassembly/reassembly operations.

3. DTPN: Dynamic Token two-level colored Petri-nets

3.1. Additional features of DTPN

Although extended two-level colored Petri-nets (XCPN) (Sakara, 2006) is the closest modeling tool to represent information for the internal control of the QRS, XCPN is not adequate to model the dynamic quality information and the quality dependent PDSP control. The characteristics of the QRS can be handled by adding the following three main supplement features to XCPN:

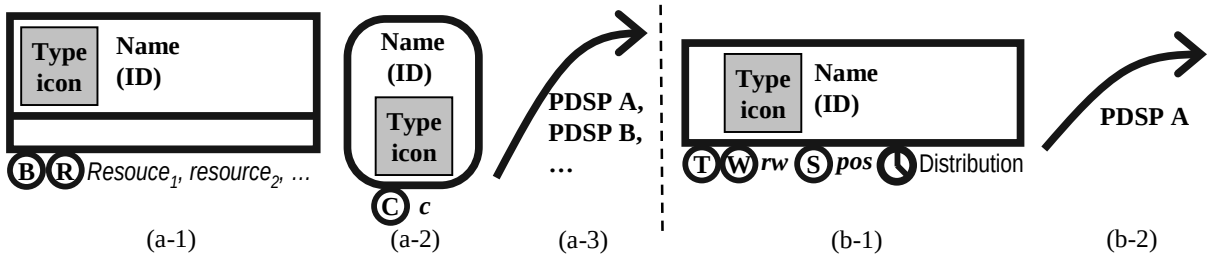


Figure VI.4. Graphical notations for the IRSR: (a) elements for the remanufacturing shop model and (b) elements for product-process model, where (a-1) workstation, (a-2) buffer, (a-3) input/output buffer and transferring PDSPs specification, (b-1) operation, and (b-2) next operation specification.

– attributes of a token can be changed dynamically;
– a token can diverge depending on the values of its own attributes;
– a transition can restrict simultaneously fire-able colors for the transition fires multiple tokens at the same time.

The first characteristic is required to embody PDSPs' dynamic information change in the model. The information change can be represented by the color feature in XCPN which is inherited from colored Petri-nets. Although a discrete value attribute of a token can be implemented by defining attributes' all possible values as different colors, the number of colors is too big to control. Besides modeling the continuous value attributes with the color feature is essentially impossible. Hence this thesis extends the color feature to represent dynamic change of the PDSP information: a color has sub-attributes in this thesis. In addition to it, DTPN also has various elements to specify the statistical distribution for the dynamic change of the values of those attributes.

The second characteristic is required to model token's dynamic routing.[1] The dynamic routing enables a token to decide the next transition to be enabled and the next places to go depending on its own attributes and other values in DTPN. These functions can model dynamic routing of a PDSP depending on its quality and system states. The conflict resolving rule[2] in XCPN can be considered as a similar feature, but it cannot reflect the token's attribute for the decision of token routing.

The last characteristic is required to model batch resources, because the simultaneously processable PDSPs by a batch resource can be restricted; some parts cannot be processed together at the same time. Therefore a transition should be able to restrict color groups which can be fired at the same time. The feature to restrict the number of simultaneously fire-able tokens also should be supported.

DTPN is an extension of XCPN and embodies the above mentioned three characteristics. This model can be utilized for the internal control of each agent in the multi-agent system. The notation Δ, ψ, and λ in the following DTPN definition are used for the statistical information, the functions for the token firing or attribute setting, and the relationships among tokens, attributes, and their possible values, respectively.

3.2. Formal definition of DTPN

This section describes only the updated and added features of DTPN compared with XCPN. In other words, this thesis does not represent non-modified features like the transition, place, arc, color, module, port, synchronization, and so on for the simplicity (refer to the appendix D for the complete definition of XCPN). DTPN is defined as follows with $\boldsymbol{PN}^{XCPN\text{-}M}$ which is the modified XCPN excluding the elements related to the transition time statistics and conflict resolving rules:

$$\boldsymbol{DTPN} = \{\boldsymbol{PN}^{XCPN\text{-}M}, \boldsymbol{A}, \boldsymbol{V}, \lambda^{pv}, \lambda^{ha}, \Delta^{ts}, \Delta^{sgt}, \Delta^{mf}, \psi^{np}, \psi^{nt}, \psi^{iv}, \psi^{sfc}\} \quad \text{(Def. VI.3.1)}$$

where each element is defined as follows:

[1] Routing means deciding a transition to be enabled or a place to receive a token by linked transition firing when a token can be fired by two or more transitions or a transition is linked with two or more output places. The decision also includes the number of tokens to be generated by transition firing.
[2] The conflict resolving rule is assigned to a place which has two or more output transitions and fires transitions as the assigned firing ratio. Refer to appendix D.

$\boldsymbol{A}$ set of attributes that colors can have; (Def. VI.3.2)

$\boldsymbol{V}$ set of value sets that attributes of colors can have; (Def. VI.3.3)

λ^{pv}: $\boldsymbol{A} \rightarrow \boldsymbol{V}$ function specifying a possible value set to an attribute; (Def. VI.3.4)

λ^{ha}: $\mathbf{CL} \rightarrow 2^{A}$ function specifying holding attributes to a color. (Def. VI.3.5)

Δ^{ts}: $\{(\boldsymbol{tr}, \boldsymbol{c}, \boldsymbol{att}, \boldsymbol{ft}_{tr}, \boldsymbol{rav}_{tr}, \boldsymbol{cr}_{tr}) \mid \boldsymbol{tr} \in \mathbf{TS} \cup (\cup \mathbf{TT_c}), \boldsymbol{c} \in \mathbf{CL}, \boldsymbol{att} \in \lambda_{ha}(\boldsymbol{c}), \boldsymbol{ft}_{tr} \in \mathbf{F}^{\mathbf{PDF}}, \boldsymbol{rav}_{tr} \in \{(\boldsymbol{av}, \boldsymbol{rto}_{av}) \mid \boldsymbol{av} \in \lambda^{pv}(\boldsymbol{att}), 0 < \boldsymbol{rto}_{av} \leq 1, \text{ and } \Sigma_{av}\, \boldsymbol{rto}_{av} = 1\}, \text{ and } \boldsymbol{cr}_{tr} \in \{(\boldsymbol{av}, \boldsymbol{cr}_{av}) \mid \boldsymbol{av} \in \lambda^{pv}(\boldsymbol{att}) \text{ and } \boldsymbol{cr}_{av} \in \mathbb{R}^{+}\}\}$

set of transition statistics for the firing time[1] distribution, the ratio of the attribute values after firing, and their correlations; (Def. VI.3.6)

Δ^{sgt}: $\{(\boldsymbol{tr}, \boldsymbol{c}, \boldsymbol{sgtf}_{tr,c}) \mid \boldsymbol{tr} \in \mathbf{TS}, \boldsymbol{c} \in \mathbf{CL}, \text{ and } \boldsymbol{sgtf}_{tr,c} \in \mathbf{F}^{\mathbf{PDF}}\}$

set of statistical distributions for the number of simultaneously generated tokens in a system-net; (Def. VI.3.7)

Δ^{mf}: $\{(\boldsymbol{trS}, \boldsymbol{trT}, \boldsymbol{mft}_{trS,trT}, \boldsymbol{mrav}_{trS,trT}) \mid \boldsymbol{trS} \in \mathbf{TS}, \boldsymbol{trT} \in \cup \mathbf{TT_c}, \boldsymbol{mft}_{trS,trT} \in \mathbb{R}^{+}, \text{ and } \boldsymbol{mrav}_{trS,trT} \in \mathbb{R}^{+}\}$

set of system-net transition modification factors by synchronizations; (Def. VI.3.8)

ψ^{np}: $\{(\boldsymbol{tr}, \boldsymbol{c}, \boldsymbol{npf}_{tr,c}) \mid \boldsymbol{tr} \in \mathbf{TS} \cup (\cup \mathbf{TT_c}), \boldsymbol{c} \in \mathbf{CL}, \text{ and } \boldsymbol{npf}_{tr,c}: \boldsymbol{PV}_c \times 2^{\mathbb{R}} \rightarrow 2^{\mathbf{PS} \cup (\cup \mathbf{PTc})}\}$

set of decision functions for tokens' next places after firing; (Def. VI.3.9)

ψ^{nt}: $\{(\boldsymbol{pl}, \boldsymbol{c}, \boldsymbol{ntf}_{pl,c}) \mid \boldsymbol{pl} \in \mathbf{PS} \cup (\cup \mathbf{PT_c}), \boldsymbol{c} \in \mathbf{CL}, \text{ and } \boldsymbol{ntf}_{pl,c}: \boldsymbol{PV}_c \times 2^{\mathbb{R}} \rightarrow \mathbf{2}^{\mathbf{TS} \cup (\cup \mathbf{TTc})}\}$

set of decision functions for tokens' next transitions to be enabled after the tokens' arrival to a place; (Def. VI.3.10)

ψ^{iv}: $\{(\boldsymbol{oc}, \boldsymbol{nc}, \boldsymbol{ivf}_{oc,nc}) \mid \boldsymbol{oc}, \boldsymbol{nc} \in \mathbf{CL}, \text{ and } \boldsymbol{ivf}_{oc,nc}: \boldsymbol{PV}_{oc} \rightarrow \boldsymbol{PV}_{nc}\}$

set of initial attribute value specifying functions to new generated tokens depending on the attribute values of a fired token; (Def. VI.3.11)

ψ^{sfc}: $\{(\boldsymbol{tr}, \boldsymbol{cc}, \boldsymbol{sfcf}_{tr,cc}) \mid \boldsymbol{tr} \in \mathbf{TS}, \boldsymbol{cc} \in 2^{\mathbf{CL}}, \text{ and } \boldsymbol{sfcf}_{tr,cc}: \{(\boldsymbol{cl}, \boldsymbol{n}_{cl}) \mid \boldsymbol{cl} \in \boldsymbol{cc} \text{ and } \boldsymbol{n}_{cl} \in \mathbb{I}^{+}\} \rightarrow \mathbb{B}\}$

set of functions specifying simultaneously fire-able color combinations and the number of fire-able tokens of each color; (Def. VI.3.12)

where

$\boldsymbol{ft}_{tr}$ firing time PDF which is assigned to a transition $\boldsymbol{tr}$; (Def. VI.3.13)

$\boldsymbol{rav}_{tr}$ occurrence frequency ratio of attribute values after $\boldsymbol{tr}$ firing; (Def. VI.3.14)

$\boldsymbol{cr}_{tr}$ correlation between firing time and attribute values after $\boldsymbol{tr}$ firing; (Def. VI.3.15)

$\boldsymbol{sgtf}_{tr,c}$ PDF of the number of simultaneously generated tokens of a color $\boldsymbol{c}$ by $\boldsymbol{tr}$ firing; (Def. VI.3.16)

$\boldsymbol{mft}_{trS,trT}$ modification factor of a transition $\boldsymbol{trS}$ for a firing time PDF $\boldsymbol{ft}_{tr}$ assigned to a synchronized transition $\boldsymbol{trT}$; (Def. VI.3.17)

$\boldsymbol{mrav}_{trS,trT}$ modification factor of $\boldsymbol{trS}$ for a attribute occurrence frequency ratio $\boldsymbol{rav}_{tr}$ assigned to synchronized $\boldsymbol{trT}$; (Def. VI.3.18)

$\boldsymbol{npf}_{tr,c}$ decision function of places to go after $\boldsymbol{tr}$ firing color $\boldsymbol{c}$ tokens; (Def. VI.3.19)

$\boldsymbol{ntf}_{pl,c}$ decision function of transitions to be enabled after the arrival of a color $\boldsymbol{c}$ token to a place $\boldsymbol{pl}$; (Def. VI.3.20)

$\boldsymbol{oc}$ color of a token which is fired by a transition;

[1] The firing time means the time elapse from a transition is enabled to the transition is fired.

$\boldsymbol{nc}$	color of a token which is generated after a transition firing;	
$\boldsymbol{ivf_{oc,nc}}$	initial attribute value decision function of a new generated color $\boldsymbol{nc}$ tokens after color $\boldsymbol{oc}$ tokens are fired;	(Def. VI.3.21)
$\boldsymbol{cc}$	combination of colors which can be simultaneously fired by a transition;	
$\boldsymbol{sfcf_{tr,cc}}$	constraint function for the simultaneously fire-able color combination $\boldsymbol{cc}$ and the number of fire-able tokens of each color by $\boldsymbol{tr}$;	(Def. VI.3.22)
TS	set of transitions in a system-net;	
PS	set of places in a system-net;	
$\mathbf{TT_c}$	set of transitions in a token-net for a color $\boldsymbol{c}$;	
$\mathbf{PT_c}$	set of places in a token-net for $\boldsymbol{c}$;	
CL	set of colors;	
$\boldsymbol{PV_c} = \{(\boldsymbol{av_{att1}}, \boldsymbol{av_{att2}}, \ldots) \mid \boldsymbol{av_{atti}} \in \boldsymbol{\lambda^{pv}}(\boldsymbol{att_i} \in \boldsymbol{\lambda^{ha}}(\boldsymbol{c}))$ and $\boldsymbol{i} = 1, 2, \ldots, \lvert\boldsymbol{\lambda^{ha}}(\boldsymbol{c})\rvert\}$		
	set of combinations of attribute's possible values of a color $\boldsymbol{c}$ token.	(Def. VI.3.23)

Here the attribute means any kind of information required to be kept in a token for the use by other functions. The dynamically changing information by transition firing are usually defined as an attribute; for example, the quality $\boldsymbol{q}$ of a PDSP and the possible quality set $\boldsymbol{Q_p}$ (def. IV.2.1) can be defined as an attribute and an attribute value set of a color defined for the PDSP.

DTPN defines the transition firing statistics set $\boldsymbol{\Delta^{ts}}$ instead of the features for transition firing time in XCPN. $\boldsymbol{\Delta^{ts}}$ enables the statistical control both of firing time and of the token's attribute value after firing, that was impossible in XCPN. $\boldsymbol{\Delta^{ts}}$ can also model the correlation between the token's attribute value change and the firing time. The statistics set for the number of simultaneously generated tokens $\boldsymbol{\Delta^{sgt}}$ can enhance the modeling-ability of $\boldsymbol{\Delta^{ts}}$. These features can be used to represent the PDSP dynamic quality change and the used product's arrival statistics. The transition statistics modification factor set $\boldsymbol{\Delta^{mf}}$ is to model the synchronization effect on the transition firing statistics. $\boldsymbol{\Delta^{mf}}$ can be utilized to model the resource quality. Tokens can be dynamically routed depending on the token's attribute values and the system states by assigning the next transition decision function $\boldsymbol{ntf_{pl,c}}$ to a place and next place decision function $\boldsymbol{npf_{tr,c}}$ to a transition. $\boldsymbol{ntf_{pl,c}}$ is called when a token arrives to a place which has two or more output transitions to decide the transitions to be enabled, and $\boldsymbol{npf_{tr,c}}$ is utilized in the same way when a transition with two or more output places. Those features are useful to model the PDSP's next operation decision function. The PDSP quality right after its disassembly/reassembly can be set with the initial attribute value decision function $\boldsymbol{ivf_{oc,nc}}$, where the color $\boldsymbol{oc}$ is for the PDSP before disassembly/reassembly and the color $\boldsymbol{nc}$ is for the PDSPs after disassembly/reassembly. Simultaneously processable PDSPs and their constraints in batch processing are modeled with the simultaneously fire-able token constraints set $\boldsymbol{\psi^{sfc}}$. Only the token set which returns true value by the constraint function $\boldsymbol{sfcf_{tr,cc}}$ can be fired simultaneously.

DTPN also requires some constraints for its soundness. An attribute's possible value set $\boldsymbol{V}$ has at least one element, because an attribute cannot have a ***null*** value (constraint VI.3.1). The transition statistics modification factor set $\boldsymbol{\Delta^{mf}}$ is to model the synchronization effect on the transition firing statistics, therefore $\boldsymbol{trS}$ and $\boldsymbol{trT}$ in an instance of $\boldsymbol{\Delta^{mf}}$ should be synchronized (constraint VI.3.2). The next transition decision function $\boldsymbol{ntf_{pl,c}}$ is only valid for the place which is linked with two or more output transitions (constraint VI.3.3), and $\boldsymbol{npf_{tr,c}}$ also

requires the same condition from the transition perspective (constraint VI.3.4). The initial attribute value decision function $\boldsymbol{ivf_{oc,nc}}$ returns attribute value of the color ***nc*** token based on the attribute values of the color ***oc*** token, hence the system-net should have a transition which generate a color ***nc*** token by firing a color ***oc*** token for the ***oc*** and ***nc*** in an instance of initial attribute value decision function set $\boldsymbol{\psi^{iv}}$ (constraint VI.3.5). The discussed constraints are defined as follows:

– $\varnothing \notin \boldsymbol{V}$; (Constraint VI.3.1)

– $\forall(\boldsymbol{trS}, \boldsymbol{trT}, \boldsymbol{mft_{trS,trT}}, \boldsymbol{mrav_{trS,trT}}) \in \boldsymbol{\Delta^{mf}}: \exists(\boldsymbol{trS}, \boldsymbol{trT}) \in \mathbf{SYN}$; (Constraint VI.3.2)

– $\forall \boldsymbol{pv} \in \boldsymbol{PV_c}: \boldsymbol{npf_{tr,c}}(\boldsymbol{pv}, 2^{R}) \cap \boldsymbol{l_{op}}(\boldsymbol{tr}) \neq \varnothing$; (Constraint VI.3.3)

– $\forall \boldsymbol{pv} \in \boldsymbol{PV_c}: \boldsymbol{ntf_{pl,c}}(\boldsymbol{pv}, 2^{R}) \cap \boldsymbol{l_{ot}}(\boldsymbol{pl}) \neq \varnothing$; (Constraint VI.3.4)

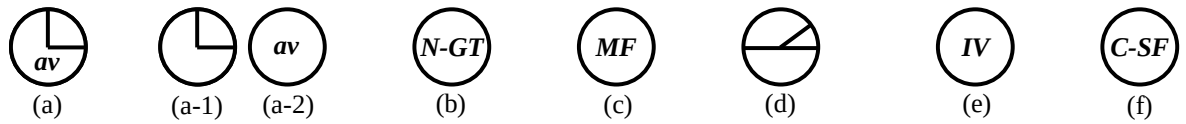
– $\forall(\boldsymbol{oc}, \boldsymbol{nc}, \boldsymbol{ivf_{oc,nc}}) \in \boldsymbol{\psi^{iv}}: \boldsymbol{nc} \in \boldsymbol{tg}_{\exists tr \in \mathbf{TS}}(\boldsymbol{oc})$, (Constraint VI.3.5)

where

SYN	set of pairs of synchronized transitions;
$\boldsymbol{l_{op}}$: $\mathbf{TS} \rightarrow 2^{\mathbf{PS}}$ or $\mathbf{TT_c} \rightarrow 2^{\mathbf{PTc}}$	function specifying linked output places to transitions;
$\boldsymbol{l_{ot}}$: $\mathbf{PS} \rightarrow 2^{\mathbf{TS}}$ or $\mathbf{PT_c} \rightarrow 2^{\mathbf{TTc}}$	function specif*ying linked output transitions of places;*
$\boldsymbol{tg_{tr}}$: $\boldsymbol{CL} \rightarrow 2^{CL}$	*function specifying an output token's color by transition* ***tr*** *firing.*

3.3. Graphical notation of DTPN

DTPN inherits all graphical notations from the parent modeling tools: basic Petri-nets and XCPN. Therefore this section defines the graphical notations only for the additional features of DTPN (refer to figure VI.5). The notation VI.5(a) is to assign the statistics information on the transition firing time and token's attribute values after firing which are in the instance of $\boldsymbol{\Delta^{ts}}$. Each statistics information can be assigned separately as notation VI.5(a-1) and VI.5(a-2). The symbols (b), (c), (d), (e), and (f) in figure VI.5 are respectively to assign the statistics of the number of simultaneously generated tokens in $\boldsymbol{\Delta^{sgt}}$, the transition firing statistics modification factors in $\boldsymbol{\Delta^{mf}}$, the token routing function in $\boldsymbol{\psi^{np}}$ and $\boldsymbol{\psi^{nt}}$, the token's initial attribute value decision function in $\boldsymbol{\psi^{iv}}$, and the simultaneously fire-able token constraint in $\boldsymbol{\psi^{sfc}}$. The assigned function names are marked right beside the rounded symbols. The attribute value sets of a color can be expressed anywhere in its token-net. Following section VI.4 shows the use of each symbol in detail.



(a) (a-1) (a-2) (b) (c) (d) (e) (f)

Figure VI.5. Graphical notations for the DTPN: a function allocation for (a) the transition time and the ratio of attributes values after firing: $\boldsymbol{ft_{tr}}$, $\boldsymbol{rav_{tr}}$, and $\boldsymbol{cr_{tr}}$, (b) the number of simultaneously generated new tokens in a system-net: $\boldsymbol{sgtf_{tr,c}}$, (c) system-net transition modification factors for synchronization effects: $\boldsymbol{mft_{trS,trT}}$ and $\boldsymbol{mrav_{trS,trT}}$, (d) the next place to go: $\boldsymbol{npf_{tr,c}}$, or the next transitions to be enabled: $\boldsymbol{ntf_{pl}}$, (e) the attributes' initial values of new tokens in a system-net: $\boldsymbol{ivf_{oc,nc}}$, and (f) the capacity calculation and constraint of simultaneously fire-able color set in a system-net: $\boldsymbol{sfcf_{tr,cs}}$.

4. Conversion methods from an IRSR model to a DTPN model

The conversion from an IRSR model to a DTPN model in this thesis means not only the expression change but also enrichment of the information for the virtual system control. A DTPN model contains more information than an IRSR model: pre-acquired knowledge about possible states and their transitions of each element type. Each element type has finite states as discussed in section V.2, hence the conversion method can be formalized by each element type in the QRS; each workstation in an IRSR model is converted into a module in a system-net of a DTPN model, the resources in the workstation occupy some parts of the module, each PDSP is converted into a token-net, and operations of the PDSP are converted into modules in the token-net. Some modules are supplemented to reflect used product arrivals and PDSP disassembly/reassembly (refer to table VI.1).

The detailed conversion method of each element type in the IRSR model is presented in the following subsections. Some IRSR features have no correspondent features in DTPN; for example, physical resource type specification function τ^{br} (def. VI.2.17) is not reflected in the DTPN model, because the human resource and machine resource have no difference in modeling the resource related part in DTPN. All the statistical information assigned to transitions in the consecutive subsections is valid only when the virtual system is used for simulations. The synchronization of the virtual system with the real-world system and the execution of the QRS deactivate the assigned statistical information; transitions are fired and tokens' attribute values change by signals from the corresponding elements in the real-world remanufacturing system during its execution.

Table VI.1. Elements in IRSR and related features in DTPN.

IRSR	DTPN	Description
M^R	System-Net	Remanufacturing shop
M^P	Token-Nets	Products and their alternative-processes
λ^{wa}	Synchronization	Workstations allocation to operations
Δ^{rq}	Δ^{mf}	Resource performance for operations time/quality
ν^{bc}	ψ^{sfc}	Batch processing restriction
W	Modules in the system-net	Workstations
R	A part of a workstation module	Resources
B	Modules in the system-net	Buffers
λ^{ra}	A part of a workstation module	Resources allocation to workstations
λ^{ib}	Module synchronization in the system-net	Input buffers of workstations
λ^{ob}	Module synchronization	Output buffers of workstations
ν^{bp}	ψ^{sfc}	Batch resource or not
ν^{cb}	Number of tokens in a place	Capacity of buffers
P	Token-nets	Alternative-processes of a part
P^U	Token-nets	Used products delivered from outside
P^R	Token-nets	Remanufactured products
λ^{d}	Module in a system-net with ψ^{np}, and ψ^{iv}	PDSPs disassembly/reassembly hierarchy
λ^{cq}	ψ^{iv}	Disassembled PDSPs qualities
Q_p	A, V, λ^{pv}, and λ^{ha}	Quality characteristics of a PDSP
O_p	Operation module in a token-net	Operations of a part
$o^s{}_p$	Synchronization and modules in a token-net	Process start operations
$o^e{}_p$	Synchronization and modules in a token-net	Process end operations
$\lambda^{no}{}_p$	Synchronization and modules in a token-net	Next operations of an operation
$\Delta^{pa}{}_p$	Module in a System-net, Δ^{ts}, Δ^{sgt}, and ψ^{iv}.	Product arrival statistics
$\Delta^{op}{}_p$	Δ^{ts}	Statistical time/quality distributions of operations
$\Delta^{cr}{}_p$	Δ^{ts}	Statistical time/quality correlations of operations
$\nu^{qt}{}_p$	A part of an operation module	Need for a quality test or not of an operation
$\nu^{rw}{}_p$	ψ^{nt}	Reworkable times of an operation
$\nu^{nor}{}_p$	ψ^{nt}	Next operation selection of an operation

* τ^{w}, τ^{br}, τ^{b}, and τ^{o} are not listed because of no correspondent features in DTPN.

4.1. Buffer

Each buffer in an IRSR model is converted into one of two kinds of modules in the system-net depending on its capacity; figure VI.6(a) and VI.6(b) respectively represent the finite capacity buffer [BF3] and the infinite capacity buffer [BF1] of the example QRS in chapter III[1]. Each transition and place has the following meanings in the QRS:[2]

T_{en} PDSP entrance into the buffer;
T_{ex} PDSP exit from the buffer;

P_c available capacity of the buffer;
P_{Pl} PDSP list contained in the buffer.

b on the arcs represents the color of the basic token to handle the buffer capacity.

The two kinds of modules are identical except for the available capacity place P_c and arcs for capacity representation. The entrance/exit of token to/from the containing PDSP list place P_{Pl} correspond to the entrance/exist of PDSP to/from the buffer. P_c in the module for a finite capacity buffer is linked one-directionally with the PDSP entrance transition T_{en} and the PDSP exit transition T_{ex} to limit the number of tokens that P_{Pl} can contain at the same time, hence the number of initial tokens in P_c and P_{Pl} has to be set as the amount of its capacity and 0 respectively; for example, P_c in the module for [BF3] should have 300 color b tokens at the initial time. But the module for an infinite capacity buffer does not require P_c, because a token can come into the buffer anytime without any restriction. Therefore infinite firing of T_{en} is possible independent of the number of tokens in P_{Pl}.

While the places in general Petri-nets usually mean the states of a modeled system, the places in the buffer module do not mean the buffer states. The places and tokens in the system-net of DTPN are considered as the facilities and PDSPs respectively. Hence the facility state can be gathered by interpreting the position of tokens. The buffer agent states (refer to figure V.7(b)) in the virtual system are decided by the number of tokens in the capacity place P_c; at least one token in P_c of the finite capacity buffer corresponds to the buffer state available, and the opposite case corresponds to the state full. The state of the infinite capacity buffer is always available.

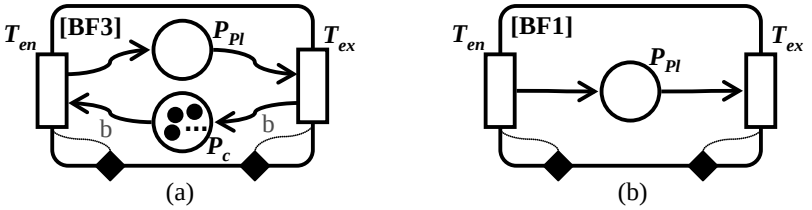


Figure VI.6. DTPN buffer modules; the finite buffer [BF3] and the infinite buffer [BF1] of the Example QRS in chapter III: (a) module for finite capacity buffer, and (b) module for infinite capacity buffer.

[1] The capacities of the buffer [BF1] and [BF3] of the QRS example in chapter III are infinite and 300 (refer to table III.1).
[2] The symbol T and P stand for a transition and a place respectively. These symbols are valid for the whole VI.4.x sections.

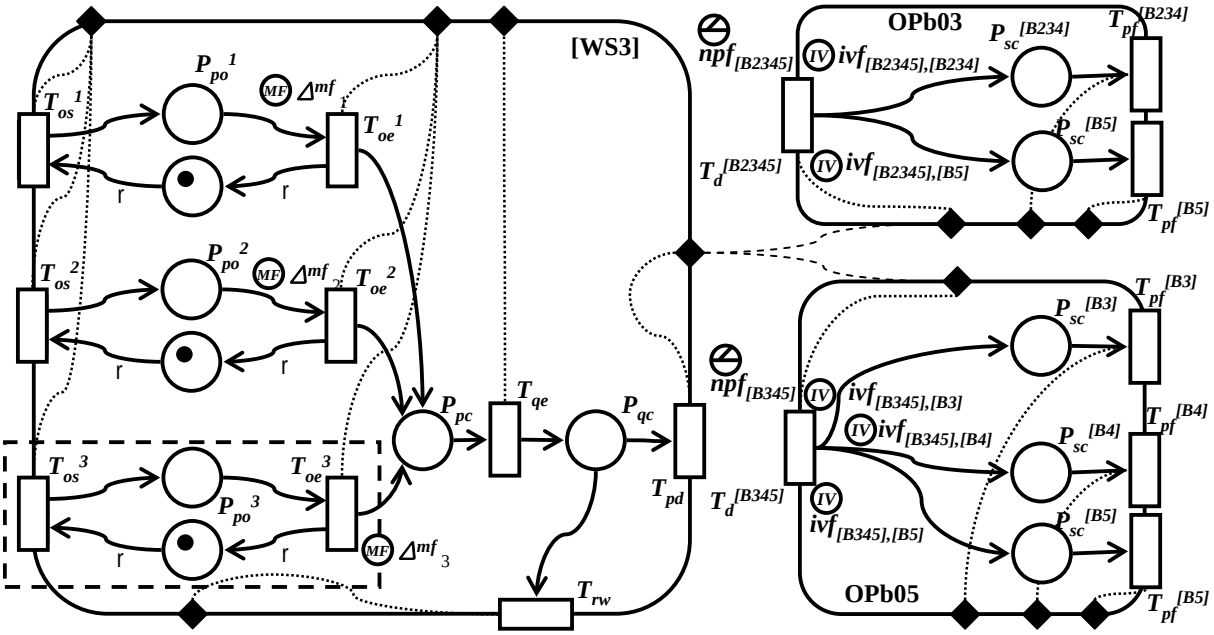


Figure VI.7. DTPN workstation module; the disassembly workstation [WS3] of the Example QRS in chapter III: (a) a module for [WS3] with three resources, and (b) supplementary modules to handle disassembly operations OPb03 (upper module) and OPb05 (lower module).

4.2. Workstation

Figure VI.7 represents a DTPN module for the disassembly workstation [WS3] with three resources of the example QRS in chapter III (refer to figures III.1 and III.5).[1] The figure only represents the case of [WS3]'s handling the disassembly operations OPb03 and OPb05 of the example used product [B].[2] Each workstation is represented with one module in DTPN. The section enclosed by dotted box in figure VI.7(a) represent resource belonged to the workstation, hence the number of the section is the same with the number of the belonging resources; the module for [WS3] in the figure has three resource sections to represents its three belonging resources. The two supplementary modules to the right of the workstation module in figure VI.7(b) are to control the PDSP disassembly in disassembly workstations. The upper and lower modules represent PDSP disassembly into two and three PDSPs by the operation OPb03 and OPb05 respectively. Each supplementary module handles one disassembly operation, hence the number of the supplementary modules is the same with the number of disassembly operations in charge; the module for [WS3] in the figure has two supplementary modules to represents the two disassembly operations OPb03 and OPb05. The number of pairs of $\boldsymbol{P_{np}}^*$ and $\boldsymbol{T_{np}}^*$ in the supplementary module is the same as the number of disassembled PDSPs. The workstation module assigns the simultaneously fire-able color constraint function $\boldsymbol{sfcf_{tr,cc}}$ (def. VI.3.22) to $\boldsymbol{T_{os}}^n$ of the module for a batch resource, which enable to handle the batch processing constraints.

Each transition and place of the three modules in figure VI.7 has the following meaning in the QRS:

[1] The subscript for the transition name of $\boldsymbol{npf_*}$ and $\boldsymbol{ivf_{*,*}}$ are omitted for the simplicity; they are marked right beside the allocated transition, hence their subscript indicating allocated transitions can be inferred without difficulty.
[2] [WS3] can also process the disassembly operation OPa04 of the used product [A] (refer to table III.4), which is omitted for the simplicity.

T_{os}^{n} operation processing start;
T_{oe}^{n} operation processing end;
T_{qe} PDSP quality examination after the operation;
T_{pd} PDSP proceeds to a next operation or is disposed of;
T_{rw} PDSP goes back to an input buffer for a rework;
T_{d}^{*} PDSP disassembled into composing PDSPs;
T_{pf}^{*} disassembled PDSP proceeds to its first operation;
P_{po}^{n} resource processes operations;
P_{pc} operation processing completion;
P_{qc} PDSP quality examination completion;
P_{sc}^{*} PDSP separation completion by disassembly.

r on the arcs in the resource section of the workstation module represents the color of the basic token to handle the resource availability. Each graphical symbol beside the statistical information or functions assigned to transitions or places are defined in figure VI.5.

Operation processing is handled by firing the operation start transition T_{os}^{n} and the operation end transition T_{oe}^{n}. T_{oe}^{n} does not have the processing time information but is synchronized with the operation processing end transition T_{pe} in the token-net (refer to figure VI.8) for the token to be fired which represents the PDSP in processing. Hence the T_{oe}^{n} firing time is basically dependent on the firing time statistics assigned to the synchronized transition T_{pe}. The T_{oe}^{n} firing time is also affected by the assigned transition modification factor set Δ^{mf} (def. VI.3.8) which contains the firing time statistics transformation factor. This approach is to handle the different operation processing time depending on not only operations but also resources in accordance with the information in the IRSR model. While the token's attributes representing the PDSP quality and the operation success or not changes by the quality examination transition T_{qe} firing during the real-world remanufacturing system control, the attributes change by the operation end transition T_{oe}^{n} firing during simulations. The QRS considers the correlation between the operation processing time and POF (refer to section IV.2.1.1), hence the T_{oe}^{n} firing time and the fired token's changed attribute should be considered at the same time. For the simplicity and convenience this thesis assigns all the operation processing related statistics information to the operation processing end transition T_{pe} in the token-net which will be discussed in the next section VI.4.3. Consequently assigning Δ^{mf} to T_{oe}^{n} without another Δ^{mf} on T_{qe} is enough to handle the resource quality affecting the operation processing time and POF. The token entered into the quality examination completion place P_{qc} is fired by either the rework transition T_{rw} or the proceed/dispose transition T_{pd} depending on the token's quality attribute value. The rework, proceed, or dispose decision function is different from PDSP to PDSP, hence the next transition decision function is not assigned to P_{qc} but to the next action decision place P_{na} in the corresponding token-net (refer to figure VI.8) as the statistical information required for T_{oe}^{n} firing.

The proceed/dispose transition T_{pd} in the disassembly workstation module should be synchronized with the disassembly transition T_{d}^{*} in the supplementary module, while T_{pd} in the non-disassembly workstation is synchronized with the input transitions T_{en} in the output buffer modules (refer to figure VI.6) of the workstation. The tokens to be created in the separation completion place P_{sc}^{*} and their initial attribute values are defined by

the next place decision function $\boldsymbol{npf}_*$ (def. VI.3.19) and the initial attribute value decision function $\boldsymbol{ivf}_{*,*}$ (def. VI.3.21) assigned on $\boldsymbol{T_d}^*$. Such approach enables the disassembled PDSP to inherit the information of the PDSP before disassembly: the due date, examined quality, and so on. New generated tokens in $\boldsymbol{P_{sc}}^*$ are immediately fired by the proceed to first operation transition $\boldsymbol{T_{pf}}^*$ and are put into output buffer modules.

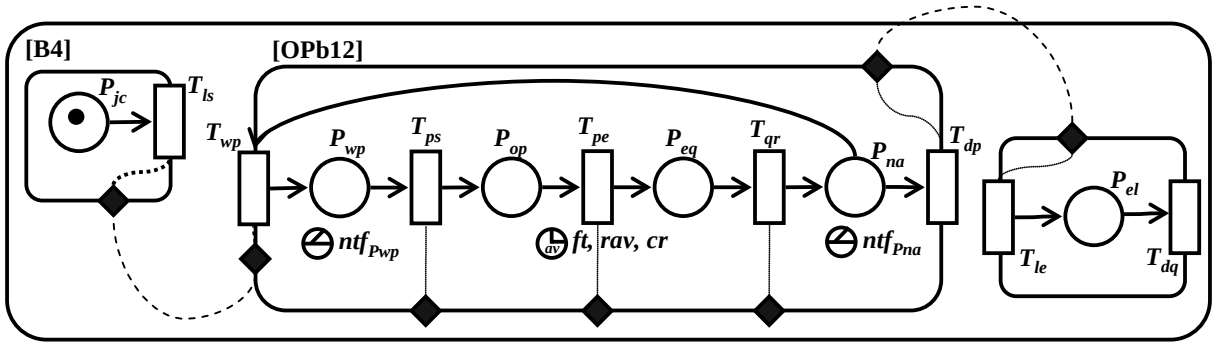
The places in the workstation and supplementary modules do not mean the states of the corresponding resource and workstation as the same with the buffer modules. Resource agent states (refer to figure V.6) in the virtual system are recognized by the tokens in the resource section of the workstation module. The state waiting for operation processing requests and the state under processing are recognized by the non-existence and existence of tokens in the operation processing place $\boldsymbol{P_{po}}^n$ respectively. The tokens to be fired are selected during the operation start transition $\boldsymbol{T_{os}}^n$ firing time, hence the resource is at the state selecting PDSPs to process during $\boldsymbol{T_{os}}^n$ firing time. Workstation agent states (refer to figure V.7(a)) in the virtual system are also recognized in the same way with that of resource states; the non-existence and existence of tokens in the operation completion place $\boldsymbol{P_{pc}}$ respectively corresponds to the state waiting for quality examination requests and the state under quality examination.

4.3. Process of the PDSP

Figure VI.8 shows a token-net for a PDSP [B4] of the example QRS in chapter III (refer to figure III.5). Each PDSP is defined as a color and has a token-net in DTPN which comprises three modules: a process start module, operation modules, and a process end module. Each operation module (refer to figure VI.8(b)) corresponds to the operation that the PDSP should undergo, hence the number of operation module in the token-net is the same with the number of operations in all alternative processes. The token-net for [B4] only has one operation module, because [B4] has one thermal cleaning operation. The process start and end modules (refer to figures VI.8(a) and VI.8(c)) are to handle the actions required before and end of processes respectively.

Each transition and place of the three modules in figure VI.8 has the following meaning of the corresponding PDSP in the QRS:

$\boldsymbol{T_{ls}}$ PDSP life start;



(a) (b) (c)

Figure VI.8. DTPN token-net; the part [B4] which has only one operation: (a) process start module, (b) operation module, and (c) process end module.

T_{wp} start to wait for operation processing;
T_{ps} operation processing start;
T_{pe} operation processing end;
T_{qr} quality information received;
T_{dp} be disposed of or proceed to a next operation;
T_{le} PDSP life end;
T_{dq} disappear from the QRS;
P_{jc} PDSP just created (appearance of PDSPs in the remanufacturing system; in other words, arrival of used products or PDSP disassembly);
P_{wp} wait operation processing;
P_{op} under operation processing;
P_{eq} under examining quality;
P_{na} deciding next action;
P_{el} ending PDSP's life.

Each graphical symbol beside the statistics information or functions assigned to transitions or places are also found in figure VI.5.

The token-net is valid right after the creation of the corresponding color token in the system-net; the used product arrival or disassembly/reassembly completion corresponds to the case. Firing the life start transition T_{ls} starts the created token's life. T_{ls} is synchronized with the operation processing wait transition T_{wp} of the first operation module. The PDSP has only one first operation in the proposed modeling tools, because IRSR adopts *virtual* operation to handle the two or more alternative first operations (refer to figure VI.2). Hence the transition and place in the process start module require no decision functions.

The token traces the places in the operation module one by one in accordance with the corresponding PDSP's operation processing. The operation processing start transition T_{ps} is synchronized with some operation processing start transitions T_{os}^{n} in the workstation modules (refer to figure VI.7), hence the token in the operation processing waiting place P_{wp} decides the transition to be enabled among the synchronized ones by the assigned next transition decision function ntf_{Pwp} (def. VI.3.20). The firing time of the operation processing end transition T_{pe} and the token's attribute values after T_{pe} firing are decided by the assigned statistical information: the firing time distribution ft (def. VI.3.13), the attribute value ratio after firing rav (def. VI.3.14), and their correlation cr (def. VI.3.15). They are also affected by the transition modification factor set Δ^{mf} (def. VI.3.8) on the operation processing end transition T_{oe}^{n} which is synchronized with T_{pe}. This modeling approach makes it possible the stochastic handling the operation processing time and the PDSP output quality in simulations. The token arrives at the next action decision place P_{na} by the examined quality information receiving transition T_{qr} firing which updates the token's quality attribute value. The token in P_{na} decides the next transition to be enabled: go back to the P_{wp} of the same operation module by the operation processing wait transition T_{wp} firing or proceed to next operations by the dispose/proceed transition T_{dp} firing, and the ntf_{Pna} deals with it. ntf_{Pna} also select to T_{wp} be enabled when T_{dp} is synchronized with T_{wp} of two or more next operation modules. This modeling is to handle the PDSP routing depending on its quality.

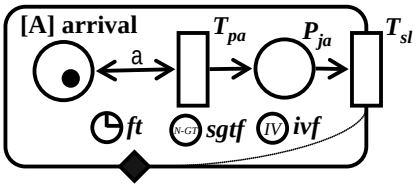


Figure VI.9. DTPN used product arrival module.

The dispose/proceed transition $\boldsymbol{T_{dp}}$ fires the life end transition $\boldsymbol{T_{le}}$ together among the synchronized transitions in case the PDSP has no more operations to process or it should be disposed, and the token goes into the process end module. The processing history of the token is collected by the control or simulation system and it disappears from the virtual system by the disappearing transition $\boldsymbol{T_{dq}}$ firing.

The places in the token-net directly represent the PDSP agent states (refer to figure V.5) in the virtual system unlike the modules for the workstation or buffer. The cases the token is in the just created place $\boldsymbol{P_{jc}}$ and the life ending place $\boldsymbol{P_{el}}$ respectively means the PDSP is in the state just created and the state ending life. The PDSP's operation processing related three states: the state waiting operation, the state under processing, and the state examining quality correspond respectively to the case the token is in the operation processing waiting place $\boldsymbol{P_{wp}}$, the under operation processing place $\boldsymbol{P_{op}}$, and the under quality examination place $\boldsymbol{P_{eq}}$. The actions at the remaining three other states: the state deciding next action, the state collecting system states information, and the state selecting operation/resource are done during the firing time of the life start transition $\boldsymbol{T_{ls}}$ and the dispose/proceed transition $\boldsymbol{T_{dp}}$. Those information collection and decision state does not affect to other elements' states, hence they are integrated and modeled as one state in the DTPN model.

4.4. PDSP creation

The token-net represents only the process of a PDSP after its appearance into the QRS. While the appearance of disassembled PDSPs is represented by supplementary modules for the workstation modules in the system-net (refer to figure VI.7(b)), the used product arrival is not modeled anywhere. Hence used product arrival control modules are required, and the module introduced in figure VI.9 deals with it. The figure represents the arrival control module for the used product [A] of the example QRS in chapter III.

Each transition and place of the three modules in figure VI.9 has the following meaning in the QRS:

$\boldsymbol{T_{pa}}$ used products arrival;
$\boldsymbol{T_{sl}}$ start used products' lives;
$\boldsymbol{P_{ja}}$ used product just arrived.

a on the arcs represents the color of the basic token to handle the used product arrival. Each graphical symbol beside the statistics information or functions assigned to transitions or places are also found in figure VI.5.

The statistical information on the used product arrival $\boldsymbol{\Delta^{pa}_{p}}$ (def. VI.2.29) is assigned together to the used product arrival transition $\boldsymbol{T_{pa}}$. The statistic information on the firing time ***ft*** (def. VI.3.13) and the number of simultaneously generated tokens ***sgtf*** (def. VI.3.16) handles the used product arrival interval and the lot size, and

the initial attribute value decision function ***ivf*** (def. VI.3.21) assigned initial quality of arrived used products. The life starting transition T_{sl} is synchronized with the life starting transition T_{ls} in the arrived used product's token-net (refer to figure VI.8(a)), hence the disappearance of created tokens from the just arrived place P_{ja} means the arrived used product started its remanufacturing process.

This module is embedded into the used product arrival manager agent (refer to figure V.8), and the agent is in the state creating PDSP agents during the firing time of the used product arrival transition T_{pa}. The agent is in the state waiting used product arrival in the other cases.

4.5. Relationship between QRS elements

Relationships among resources, buffers, and PDSPs are represented with the synchronization feature of DTPN. The relationship between PDSP and workstations are modeled by synchronizing each transition of the operation modules in the token-net and the workstation modules in the system-net (refer to figures VI.7 and VI.8, and sections VI.4.2 and VI.4.3); for example, the operation processing start transition T_{ps} of the operation module in the token-net is synchronized with the operation processing start transitions T_{os}^{n} of the workstation modules in the system-net which can process the operation. The workstation allocation to operations function λ^{wa} (def. VI.2.41) can be modeled by this approach.

The transitions in the buffer modules and workstation modules are synchronized to handle the PDSP's transference in accordance with the operation processing (refer to figures VI.6 and VI.7); the exit transition T_{ex} of the buffer module in the system-net is synchronized with T_{os}^{n}, and the entrance transition T_{en} of the buffer module is synchronized with the proceed/dispose transition T_{pd} or the proceed to the first operation transition T_{pf}^{*}; T_{pd} in the non disassembly workstation is directly synchronized with T_{en}, because it requires no supplementary modules to handle disassembly.

5. Validation of the QRS modeling tools

5.1. Distinctive features compared to previous modeling tools

This thesis proposed the two new modeling tools because of no suitable tools for the QRS modeling: IRSR and DTPN. The proposed tools are motivated by existing modeling tools: disassembly Petri-nets (Zussman and Zhou 1999) and XCPN (Sakara 2006), hence this section discusses the improved features compared to the previous modeling tools.

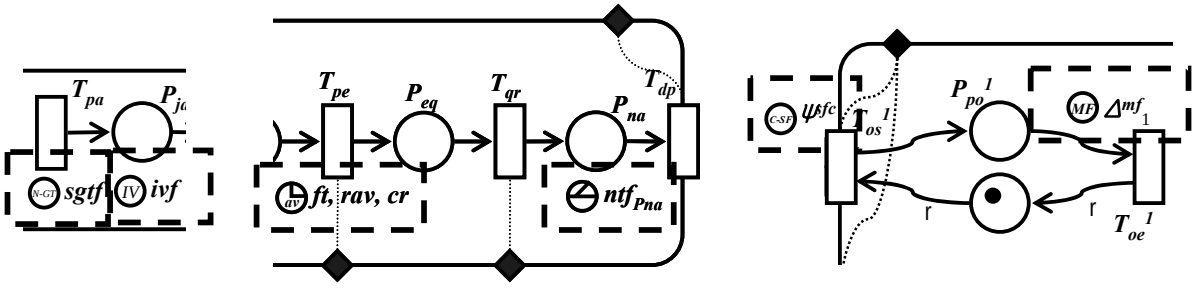
The disassembly Petri-nets (refer to section II.1.1) and XCPN are extended Petri-nets represented with the concept of states management. Therefore the remanufacturing system models designed with those tools are not easy to understand for the field technicians. Furthermore no guidelines for the remanufacturing system design cause inconsistency among models (refer to section V.1.2.1). To cope with the difficulty and not to corrupt the Petri-nets framework, IRSR is proposed with the conversion method from an IRSR model to a DTPN model. IRSR is easy to design and understand even for the field technicians who have no knowledge on the Petri-nets and multi-agent, because it comprises basic elements in the remanufacturing system: workstations, buffers, and

operations. IRSR embeds the quality representation (refer to section IV.2): the PDSP's possible quality set Q_p (def. IV.2.1) and the resource quality Δ^{rq} (def. VI.2.42). It also supports the next operation decision functions $f^{nor}{}_o$ (def. VI.2.39) which can be applied after each operation processing completion. Those features enable to model the PDSP/resource quality and the quality dependent operation processing time and POF and to select the remanufacturing process dynamically.

DTPN has some new features (refer to section VI.3.1): the dynamically changeable token attribute, the dynamic token routing depending on each token's attribute values, the different firing time depending on the synchronized transition combinations, the simultaneous one or more tokens generation by statistics information, and the constraints for the simultaneously fire-able tokens. Each feature respectively enables to model the dynamic quality representation, the dynamic remanufacturing process selection depending on the PDSP quality and QRS states, the resource quality dependent operation processing time and output PDSP quality, the used product's bulk arrival, and the batch processing constraints. Actually XCPN can model the third and fourth features; define as many operation modules as the number of resources which can process the operation and define as many transitions as the number of possible lot size of used product's bulk arrival. But such modeling approach results in a complicated model, hence the integrated modeling with the proposed features is preferable for simplicity. Other three features are essentially impossible to model with the previous modeling tools. The use of new features in the remanufacturing system modeling with DTPN is represented in figure VI.10 which comprises the extracted parts from the used product arrival, operation, and workstation modules introduced in the previous section VI.4. This section omits the explanation on the assigned functions and statistics information, because each subsections of section VI.4 explains them in detail.

The distinctive features of IRSR and DTPN can be synthesized as follows:

– Intuitively understandable user-side modeling approach;
– Structured definite system-side modeling approach for each element in the QRS;
– Quality representation of the PDSP and resource;
– Dynamically changing attributes of the token in DTPN;
– Token routing depending on token's attribute values and system states in DTPN;



(a) (b) (c)

Figure VI.10. Graphical notations for the new characteristics of DTPN compared to the previous modeling tools: (a) the stochastic simultaneous generation of one or more tokens and the attributes initial values setting (a part of the product arrival module in figure VI.9), (b) the dynamic attributes values change and the token routing depending on token's attribute values and system states (a part of the operation module in figure VI.8), and (c) the restriction on simultaneously fire-able tokens and the statistics information modification for synchronized transitions (a part of the workstation module in figure VI.7).

– Different firing time statistics depending on the synchronized transition combination;
– Constraints of the simultaneously fire-able tokens in DTPN;
– Stochastic generation of multiple tokens at one time in DTPN.

5.2. Comparison with previous modeling tools by simulations

This section numerically compares the proposed modeling tools with XCPN to indirectly show that IRSR/DTPN covers all features of XCPN and can emulate the XCPN model. DTPN inherits all features of XCPN, consequently DTPN theoretically covers XCPN. But some features like the conflict resolving rule[1] is updated with new feature: the next transition decision function $\boldsymbol{ntf_{pl,c}}$ (def. VI.3.20), hence the numerical comparison is required to confirm the consistency of the two modeling approaches.

System performance of the example QRS in chapter III are evaluated for two cases: the IRSR/DTPN model and the emulated XCPN model. The XCPN model is emulated by eliminating the following two features which cannot be modeled with XCPN:

– The operation processing time and POF difference depending on resources and input PDSP quality;
– The operation/resource selection functions depending on PDSP/resource quality and system states.

Hence the emulated XCPN model contains the same information with XCPN model for the example QRS. Each of the operation processing time and POF should be integrated into the one statistics independent of the resource/operation combinations to eliminate the above two features; those values are gathered by simulations of the IRSR model. Appendix E represents the complete IRSR model of the example QRS in chapter III, and appendix G represents the processing time and POF part of the emulated XCPN model constructed with the operation processing time and POF gathered by simulations.

Performance of the IRSR/DTPN model and the emulated XCPN model are evaluated by 10 times simulations where the total work content (TWK) approach with the adjustment factor 3.58 is adopted for due date setting.[2] 5,000 used products are handled for each simulation, resources/operations are randomly selected, batch resources instantly start operation processing even at least one PDSP is in the waiting queue, and the first come first serve (FCFS) rule is applied to all resource to dispatch PDSPs. The detailed experiment design for simulations will be explained later in section VII.2.2.

Simulation results are summarized in table VI.2, where the values are the average of 10 times simulations. The mean and standard deviation (STD) of flow time show respectively 0.38% and 0.93% differences between two models (refer to the third row of table VI.2). The tardiness measures also have under 1% differences; the differences of the mean and STD are 0.49% and 0.85% respectively, and the percentage of tardy jobs show only 0.02% difference. Even though the maximum flow time and tardiness have comparably big differences of 3.10% and 2.31% respectively, it is natural outcome because of the characteristic of the maximum value: the value is dominated by just one abnormal worst case.

[1] The conflict resolving rule fires alternative transitions just by the assigned transition enabling ratio without any other information.
[2] The TWK adjustment factor is gathered by pilot simulation; 5 times with 3,000 used products.

Table VI.2. Performance of the example QRS by modeling with IRSR and the emulated XCPN.

Model	Flow time (minutes)			Tardiness (minutes)			
	Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[a] (%)
IRSR $^{(\alpha)}$	523.1	2,316	425.6	346.6	2,197	436.3	80.7
Emulated XCPN $^{(\beta)}$	525.1	2,388	429.5	348.3	2,248	440.0	80.7
Difference[b] (%)	0.38	3.10	0.93	0.49	2.31	0.85	0.02
P-value from t-test	0.97	0.77	0.95	0.97	0.83	0.95	0.99

[a] PoTJ: Percentage of tardy jobs.
[b] [difference] = $|\alpha-\beta|/\boldsymbol{min}(\alpha, \beta)\times 100$; the calculation results can be different with the value calculated with that in the table because of the value in the table is rounded from the original value.

The Student's t-test[1] is done to make sure the nondiscrimination of the two models (refer to the fourth row of table VI.2); independent unequal distribution is assumed because of the independency among each simulation, and the one-tailed test is done to examine if they have different distributions or not. The t-test results show that the mean and STD of two measures are not different in 97% and 95% confidence level respectively and that the percentage of tardy jobs is not different in 99% confidence level. Hence this thesis can concludes that the simulation results from IRSR and the emulated XCPN are not different. Consequently IRSR can be considered indirectly to cover all features of XCPN and be also utilized for the modeling of remanufacturing systems without quality information.

[1] This thesis utilized the Microsoft® office excel 2003 for all the mathematical and statistical calculations.

VII. Quality Embedded Dispatching Rules

This chapter proposes dispatching rules which will be embedded into the knowledge-based scheduling mechanism (refer to chapter VIII). The quality related factors utilized in the proposed dispatching rules are the operation processing time and the probability of operation success (POS) which are different depending on the resource quality and the used product and disassembled subassembly/part (PDSP) quality. Hence the rules can be utilized only when the QRS maintains the statistical information of operation processing discussed in chapter IV.

Abbreviated terms used in this chapter:

EDD	Earliest Due Date;
FCFS	First Come First Serve;
LPOS	Lowest POS;
LPOSD	LPOS to Due date;
LPOSM	LPOS to Modified due date;
LPOSR	LPOS to Remaining total processing time;
LPOST	LPOS to operation processing Time;
LPOSU	LPOS to due date Urgency;
MDD	Modified Due Date;
NEDD	New EDD;
PDSP	used Product and Disassembled Subassembly/Part
POF	Probability of Operation Failure;
POS	Probability of Operation Success;
QRS	Quality embedded Remanufacturing System;
RPT	Remaining Processing Time;
SPT	Shortest Processing Time;
STD	STandard Deviation;
TWK	Total Work content;
WINQ	Work In Next Queue.

1. New dispatching rules

This thesis proposes six quality embedded dispatching rules. The basic dispatching rule for all of them is the lowest probability of operation success (LPOS) rule. The priority calculation function of the LPOS rule for the operation ***o*** which a PDSP ***p*** with input quality ***iq*** requested to a resource ***r*** is defined as follows:

$$pri^{LPOS}_{r,o,iq} = 1 - pof_{o,iq} \times cf^{r-POF}_{o,r} \quad = 1-[\text{POF}]\times[\text{resource quality to the POF}]. \qquad \text{(Def. VII.1.1)}$$

where

$pof_{o,iq}$ POF (def. IV.2.7);
$cf^{r\text{-}POF}_{o,r}$ resource quality factor to the POF (def. IV.2.18).

The formula reflects the resource quality effect to the POS. This thesis defines the resource quality factor to the probability of operation failure (POF) $cf^{r\text{-}POF}_{o,r}$ instead of that to the probability of operation success (POS), hence resource quality reflected POF should be calculated first and converted into the POS. The LPOS rule dispatches first the lowest success probability operation. Therefore the waiting PDSPs are ordered by their requesting operation' POSs. The PDSP getting the minimum value by the formula has the highest priority. The following other calculation functions for the proposed dispatching rules are also applied in the same way; a smaller value by the formula means a higher priority.

The LPOS rule causes the waiting time of the lower POS PDSPs to decrease and that of higher POS PDSPs to increase. Hence the PDSP flow time variance is expected to decrease, because the processing times of lower POS PDSPs are bigger than that of higher POS PDSPs because of their more rework. The flow time decrease can be also expected by LPOS from the perspective of the immediate PDSP replacement; a disposed PDSP is immediately substituted with a new one which is directly sent to the reassembly workstation, and the reassembly operation waiting for the PDSP can be processed. Hence the reassembly operation can start earlier than the waiting PDSP's non disposal and refurbishment case and save the waiting time as the refurbishment time.

This thesis defines five more dispatching rules based on the LPOS rule as follows: the lowest POS to operation processing time (LPOST) rule, the lowest POS to due date urgency (LPOSU) rule, the lowest POS to remaining total processing time (LPOSR)rule, the lowest POS to due date (LPOSD) rule, and the lowest POS to modified due date (LPOSM) rule:

$$pri^{LPOST}_{r,o.iq} = pri^{LPOS}_{r,o.iq} \times avg\left(pt_{o,iq} \times cf^{r-pt}_{o,r} \right)$$

= [resource quality reflected POS]×***avg***([operation processing time]
×[resource quality to the operation processing time]) ; (Def. VII.1.2)

$$pri^{LPOSU}_{r,p,o.iq} = pri^{LPOS}_{r,o.iq} \times (t^{d}_{p} - t^{current})$$

= [resource quality reflected POS]×([due date]−[current time]); (Def. VII.1.3)

$$pri^{LPOSR}{}_{r,p,o.iq} = pri^{LPOS}{}_{r,o.iq} \times t^{rp}{}_{p}$$

= [resource quality reflected POS]×[remaining processing time]; (Def. VII.1.4)

$$pri^{LPOSD}{}_{r,p,o.iq} = pri^{LPOS}{}_{r,o.iq} \times t^{d}{}_{p}$$

= [resource quality reflected POS]×[due date]; (Def. VII.1.5)

$$pri^{LPOSM}{}_{r,p,o.iq} = pri^{LPOS}{}_{r,o.iq} \times max(t^{d}{}_{p}, t^{current} + t^{rp}{}_{p})$$

= [resource quality reflected POS]

×***max***([due date], [current time]+[remaining processing time]), (Def. VII.1.6)

where

$pri^{LPOS}{}_{r,o,iq}$ priority calculation function of the LPOS rule (def. VII.1.1);

$pt_{o,iq}$ operation processing time statistics (def. IV.2.3);

$cf^{r\text{-}pt}{}_{o,r}$ resource quality factor to the operation processing time statistics (def. IV.2.17);

$t^{d}{}_{p} \in \mathbb{T}$ due date of a PDSP ***p***; (Def. VII.1.7)

$t^{current} \in \mathbb{T}$ current time, in other words, the time instance of the dispatching rule application. (Def. VII.1.8)

$t^{rp}{}_{p} \in \mathbb{T}$ remaining remanufacturing processing time of a PDSP ***p*** at current time. (Def. VII.1.9)

The above defined rules are combinations of the LPOS rule and existing popular conventional dispatching rules like the shortest processing time (SPT) rule except for the LPOSU rule. The combined rules with LPOS are widely used because of their simplicity and effectiveness in certain system states. LPOST, LPOSR, LPOSD, and LPOSM combine the LPOS rule with the shortest processing time (SPT), remaining processing time (RPT), earliest due date (EDD), and modified due date (MDD) rules respectively. This thesis call each of combined dispatching rule as mother dispatching rule; for example, the SPT rule is the mother dispatching rule of the LPOST rule. The LPOSU rule combines LPOS with the operation urgency by calculating the remaining time to the due date.

Both the parent dispatching rules and LPOS give the higher priority to the PDSPs having the smaller value by their calculation formulas. Therefore the priority calculation function of each proposed dispatching rule is derived by multiplying formulas from the two combined element dispatching rules to keep their original characteristics. The proposed dispatching rules are expected to show better flow time than their parent dispatching rules because of the LPOS's waiting time reduction effect discussed above. The next section examines the performance of the six proposed dispatching rules by comparing with popular conventional dispatching rules in the manufacturing/remanufacturing system.

2. Performance of proposed dispatching rules

This thesis examines the performance of the above proposed six dispatching rules with the example QRS in chapter III. [1] Performance is usually examined under the busy and idle system cases in dispatching rule

[1] Refer to appendix E for the details of the numerical values of all input parameters required for simulations in this chapter.

comparisons. Hence the used product arrival information in the example QRS is modified; the used product arrival lot size distribution is adjusted to generate such cases. Performance is compared without using any scheduling mechanism to examine the net performance of the dispatching rules themselves; in other words, PDSPs randomly select operations/resources and batch resources immediately start processing whenever they have at least one PDSP in their waiting queue.

2.1. Benchmark dispatching rules

This thesis compares the performance of the proposed six dispatching rules with seven other conventional popular dispatching rules (refer to table VII.1). Six of them are used as the comparative rules in many previous publications, and the new earliest due date (NEDD) rule is a good performance dispatching rule for the remanufacturing system (Sakara 2006). The variables in table VII.1 are defined as follows:

$pt_{o,iq}$ operation processing time statistics (def. IV.2.3);
$cf^{r\text{-}pt}_{o,r}$ resource quality factor to the operation processing time statistics (def. IV.2.17);
t^{rp}_{p} remaining remanufacturing processing time at current time (def. VII.1.9);
t^{awq}_{p} arrival time to the waiting queue of a PDSP p; (Def. VII.2.1)
t^{d}_{p} due date (def. VII.1.7);
$t^{current}$ current time (def. VII.1.8);
n^{sdP}_{p} number of PDSPs in the remanufacturing system which have the same due date with a PDSP p;(Def. VII.2.2)
d^{min} minimum value of due dates of PDSPs in the remanufacturing system at current time; (Def. VII.2.3)
$p^{rv}_{o,iq,r}$ probability of visiting a resource r after an operation o with input quality iq; (Def. VII.2.4)
pt^{wq}_{r} sum of the operation processing time of the PDSPs in the waiting queue of a resource r. (Def. VII.2.5)

Table VII.1. Selected dispatching rules for the performance comparison with the proposed dispatching rules.

Category	Dispatching rule name	Priority: higher priority	Priority: Calculation function
Flow time	Shortest Processing Time (SPT)	shorter processing time of the operation to be processed	$avg(pt_{o,iq})\times cf^{r\text{-}pt}_{o,r}$
	Remaining Processing Time (RPT)	shorter processing time of the whole remaining process	t^{rp}_{p}
	First Come First Served (FCFS)	earlier arrived to the waiting queue	t^{awq}_{p}
Due date	Earliest Due Date (EDD)	earlier due date	t^{d}_{p}
	Modified Due Date (MDD)	earlier modified due date which is bigger value between the due date and the current time plus the total remaining processing time	$max(t^{d}_{p}, t^{current} + t^{rp}_{p})$
	New Earliest Due Date (NEDD)	larger number of jobs in the same due date group and earlier due date	$\exp\left(-n^{sdP}_{p} \Big/ 1 + t^{d}_{p} - d^{min}\right)$
Work in process	Work In Next Queue (WINQ)	smaller works (total processing time) in the waiting queue of resources to visit after the operation processing	$avg(p^{rv}_{o,iq,r} \times pt^{wq}_{r})$

2.2. Experiment design

The performance of the manufacturing/remanufacturing system is usually examined by small trials with large volumes (Law and Kelton 1984 and Sakara, 2006). Many publications recognize that about 10 trials with the order of thousands jobs for each trial are enough. Hence this thesis simulated 10 times for each dispatching rule application case, and each simulation was executed up to 5,000 used products which are collected and all of them are remanufactured. An identical dispatching rule is applied to all resources for each simulation; for example, the simulation for the SPT rule means the simulation under the SPT rule application to all resources. This thesis executes some pilot simulations to find simulation parameters for main simulations; [1] for example, finding due date adjustment factors which will be explained later. The pilot simulation is done with 5 times simulations with 3,000 used product arrival and remanufacturing completion.

Many publications examined their approaches under two kinds of resource utilization rates: 75% as the low resource utilization rates and 95% as the high; this thesis calls each of those system states as the idle and busy systems respectively. The resources are fixed facilities, hence this thesis adjusts the used product arrival lot size statistics to make the different system states. The 100% resource utilization of the remanufacturing system having one resource means the used product arrival interval is equal to the whole processing time, hence the resource utilization rate decrease as the more resources in the system and the less lot size at each used product arrival; consequently the relationships among resource utilization rate, used product arrival interval and lot size, and remanufacturing processing time for the one resource case are as follows:

$$[\text{resource utilization rate}] = [\text{remanufacturing processing time}] \Big/ \frac{[\text{used product arrival interval}]}{[\text{used product lot size at each arrival}]}$$

$$\Leftrightarrow [\text{used product lot size at each arrival}] = \frac{[\text{used product arrival interval}] \times [\text{resource utilization rate}]}{[\text{remanufacturing processing time}]}.$$

Based on the idea is derived the following corresponding lot size $\boldsymbol{ls^c_{up}}$ calculation formula of the arrived used products $\boldsymbol{up}$:

$$\boldsymbol{ls^c_{up}} = \boldsymbol{avg}(\boldsymbol{ls_{up}}) = \frac{\boldsymbol{avg}(\boldsymbol{ai_{up}}) \times \boldsymbol{r}^{\boldsymbol{ru}}}{\boldsymbol{max}_{r \in R}\left(\sum_{o \in O_r} \left(\frac{\boldsymbol{avg}(\boldsymbol{pt_{o,NDQ}}) \times (\boldsymbol{1} + \boldsymbol{pof_{o,NDQ}}) \times \boldsymbol{p}^{\boldsymbol{op}}{}_{\boldsymbol{o}}}{\boldsymbol{c_{r,o}} \times \sum_{w \in \boldsymbol{\lambda}^{\boldsymbol{wa}}(o)} |\boldsymbol{\lambda}^{\boldsymbol{ra}^{-1}}(\boldsymbol{w})|} \right) \right)}$$

$$= \frac{[\text{expected arrival interval}] \times [\text{resource utilization rate}]}{\boldsymbol{max}_{r \in R}\left(\sum_{o \in O_r} \left(\frac{[\text{rework considered average processing time of } \boldsymbol{o}] \times [\text{probability of } \boldsymbol{o} \text{ processing}]}{[\text{number of resources capable of } \boldsymbol{o} \text{ processing}]} \right) \right)}$$

$$= \frac{[\text{expected arrival interval}] \times [\text{resource utilization rate}]}{\boldsymbol{max}_{r \in R}\left([\text{expected utilization time of } \boldsymbol{r} \text{ for one used product remanufacturing }] \right)}$$

[1] Simulations usually required much time, hence pilot simulations are executed in case the simulation results are not directly used for the system performance evaluation.

$$= \frac{[\text{expected arrival interval}] \times [\text{resource utilization rate}]}{[\text{maximum utilization time among resources for one used product remanufacturing }]}, \quad \text{(Def. VII.2.6)}$$

where

$\boldsymbol{ls_{up}}$ ($\boldsymbol{ls_p}$)	lot size of each used product bulk arrival (def. VI.2.37);	
$\boldsymbol{ai_{up}}$ ($\boldsymbol{ai_p}$)	arrival interval statistics (def. VI.2.36);	
$\boldsymbol{r^{ru}}$	resource utilization rate;	(Def. VII.2.7)
$\boldsymbol{R}$	set of resources (def. VI.2.9);	
$\boldsymbol{O_r}$	set of operations which can be processed by the resource $\boldsymbol{r}$;	(Def. VII.2.8)
$\boldsymbol{pt_{o,iq}}$	operation processing time statistics (def. IV.2.3);	
$\boldsymbol{pof_{o,iq}}$	POF (def. IV.2.7);	
$\boldsymbol{p^{op}_o}$	probability of operation $\boldsymbol{o}$ processing;	(Def. VII.2.9)
$\boldsymbol{c_{r,o}}$	simultaneous PDSP processing capacity of the resource $\boldsymbol{r}$ for the operation $\boldsymbol{o}$.	(Def. VII.2.10)
$\boldsymbol{\lambda^{wa}}$	workstation allocation function (def. VI.2.41);	
$\boldsymbol{\lambda^{ra}}$	belonging workstation specification function (def. VI.2.11);	

The ***NDQ*** in the input quality place of symbols means the no defect PDSPs quality; this thesis uses the operation processing time and POS of the no defect PDSP input case as the representative among various operation processing time and POS information depending on the PDSP input qualities. This thesis considers only one rework by the operation failure to calculate the expected operation processing time; the probability of the second or more rework is very small, because the square, cube, or more of POF is very small. The probability of operation processing $\boldsymbol{p^{op}_o}$ is calculated by dividing the number of the operation's belonging alternative processes with the number of all possible alternative processes; for example, the $\boldsymbol{p^{op}_o}$ is 0.75 (= 3/4) when an operation $\boldsymbol{o}$ belongs to the three remanufacturing processes among four alternative possible remanufacturing processes. The simultaneous PDSP processing capacity $\boldsymbol{c_{r,o}}$ for non-batch resources are always 1, and $\boldsymbol{c_{r,o}}$ for a batch resource should be gathered from the batch processing constraint function $\boldsymbol{f^{bc}_{r,oB}}$ (def. VI.2.45). Table III.7 describes the batch capacity of the example QRS. This thesis considers maximum utilization rate among resources as the resource utilization rate of the system, hence the real resource utilization rate is lower than the target utilization rate. This approach is unavoidable for the remanufacturing system where the operations cannot equally share the resources and the used product arrival frequency exceeds the processing capacity of at least one resource which is called as a bottle neck resource. Such remanufacturing system makes the other resources wait until the bottle neck resource completes its operation, especially the resources in charge of the operations to be done after the operation handled by the bottle neck resource. Consequently the other resources' waiting decreases the resource utilization rate of the whole system.

The arrival lot sizes of used products [A] and [B] in the example QRS are 27.3 and 25.0 for the busy system and 21.6 and 19.7 for the idle system respectively. They are calculated with the above formula and divided by two, because they share an identical remanufacturing shop at the same time. The example QRS defines the arrival lot size as a normal distribution with standard deviation (STD) of 10% and 20% for each product [A] and [B] respectively (refer to table III.5), and the distribution type and STD percentage are also maintained in the busy and idle system cases.

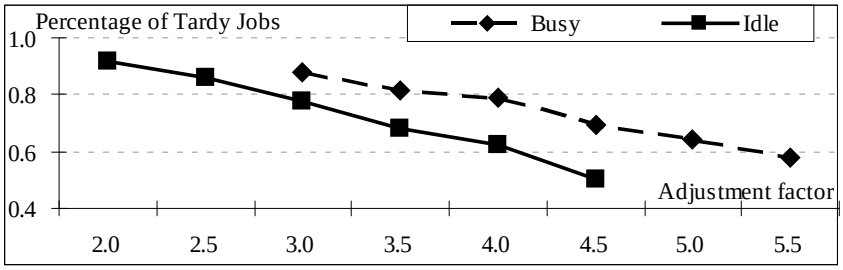


Figure VII.1. Percentage of tardy jobs depending on due date adjustment factors.

The performance of dispatching rules is different depending on the applied due date assignment method (Baker 1984), and many researchers usually accept the total work content (TWK) method because of its simplicity and rationality (Holthaus and Rajendran 1997). The TWK assigns a due date to an arrived job to the system with the following simple calculation:

$$c^{df} \times t^{tw},$$

where

c^{df} due date adjustment factor;

t^{tw} sum of processing time of work contents.

The sum of processing time of work contents t^{tw} can be considered as the whole remanufacturing processing time of a used product. But the collected used products have different processes depending on their quality, hence this thesis uses the average processing time of all possible alternative processes. Many previous research set the value of due date adjustment factor c^{df} by pilot simulation to meet the 75% tardy jobs level, and they are usually accepted as the appropriate tardiness level to compare dispatching rules (Sakara 2006). Figure VII.1 represents the percentage of tardy jobs depending on c^{df} by pilot simulations with the FCFS rule. c^{df} for the 75% tardy jobs are 4.20 and 3.13 for the busy and idle systems respectively.

2.3. Simulation results and analysis

This section discusses the simulation results for dispatching rule application cases. Values for each performance measure are normalized by the minimum value of the measure among the compared dispatching rules; for example, LPOSU shows the smallest mean flow time of 574.3 minutes in the busy system, then 1.025 mean flow time of MDD indicates a 2.5% bigger value as 588.7 (= 574.3×1.025) (refer to figure VII.2). This section discusses dispatching rules with the traditional popular performance measures: flow time and tardiness, and the performance of a dispatching rule means the system performance when the dispatching rule is applied.

2.3.1. Busy system

LPOSU is generally the best for both the flow time and tardiness in the busy system, and LPOSM and MDD follows it.

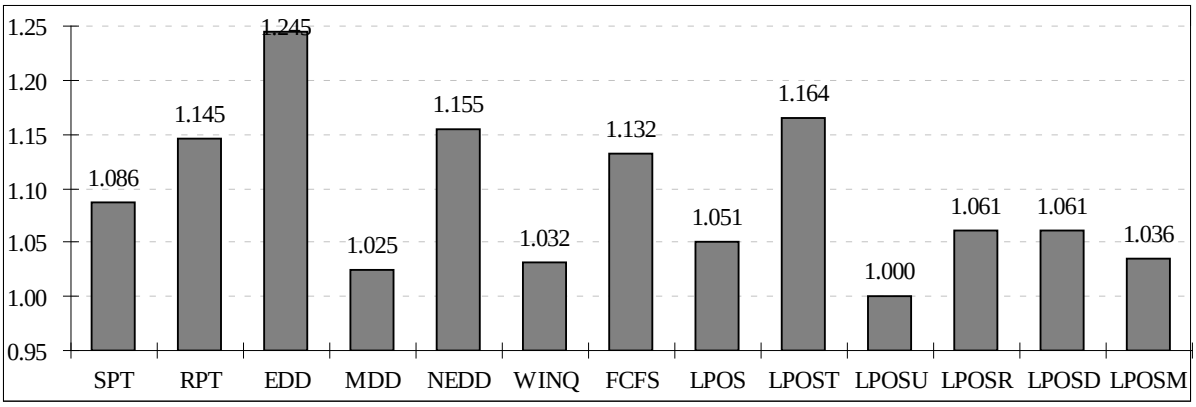


Figure VII.2. Mean flow time in the busy system case.

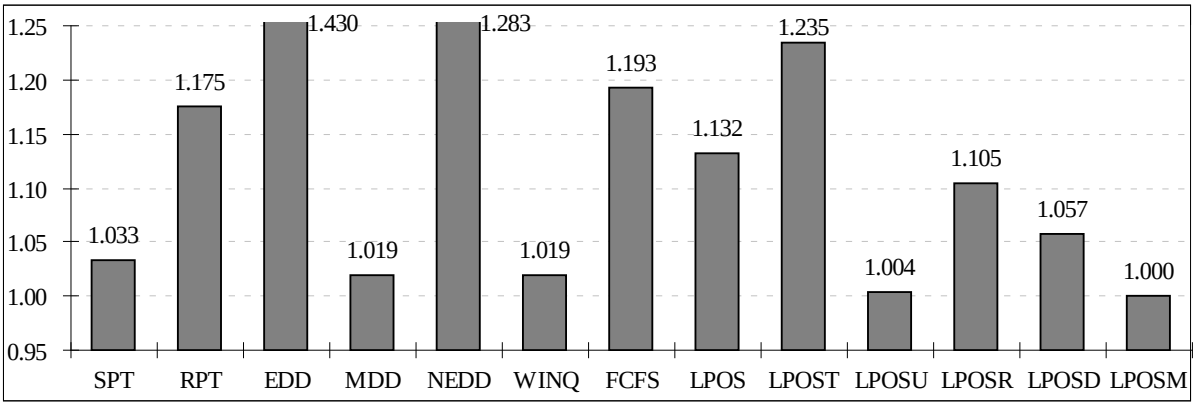


Figure VII.3. Flow time STD in the busy system case.

Figure VII.2 represents that LPOSU shows about 3% better mean flow time than other dispatching rules in the second better performance group; MDD, WINQ, and LPOSM which show 2.5%, 3.2%, and 3.6% worse performances respectively. The above dispatching rules showing better mean flow time also show better flow time STD, but the sequence is different; LPOSM is the best, and LPOSU, MDD, and WINQ follows it (refer to figure VII.3). Their performance difference in the flow time STD is comparably smaller than that in the mean flow time; LPOSM is only 0.4% better than LPOSU and 1.9% better than MDD and WINQ. The performance relationship between each proposed rule and its parent dispatching rule show no consistency, and no performance improvement by dispatching rule combination is found; LPOSR and LPOSD respectively show better mean flow time than their mother dispatching rules of RPT and EDD, but LPOSM and LPOST are contrary. The LPOST rule even shows 7.2% (= 1.164/1.086l) and 10.8% (= 1.164/1.051−1) worse mean flow time than both of its elementary dispatching rules: SPT and LPOS respectively.

Tardiness measures are usually highly synchronized with the flow time measures, the longer the average flow time is, the more the tardy jobs are. Figures VII.4 and VII.5 represent that LPOSU and LPOSM respectively show the best performance in mean tardiness and tardiness STD as the flow time measures. LPOSM and MDD follow the best performance dispatching rule in the mean tardiness, and LPOSU, MDD, and WINQ follows it in the tardiness STD. The proposed dispatching rules also does not show a definite performance improvement; LPOSM is better than MDD but only 2.0% (= 1.033/1.054), and LPOST still shows worse performance than its elementary dispatching rules.

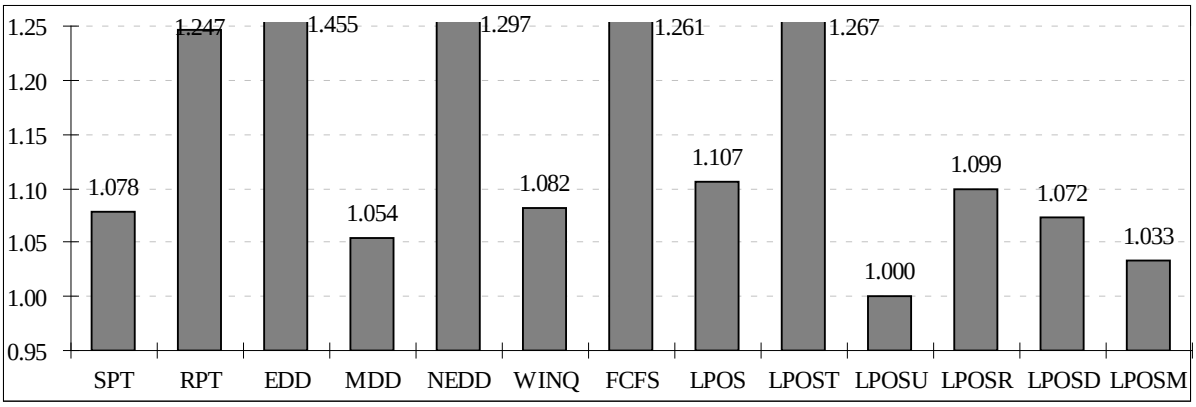


Figure VII.4. Mean tardiness in the busy system case.

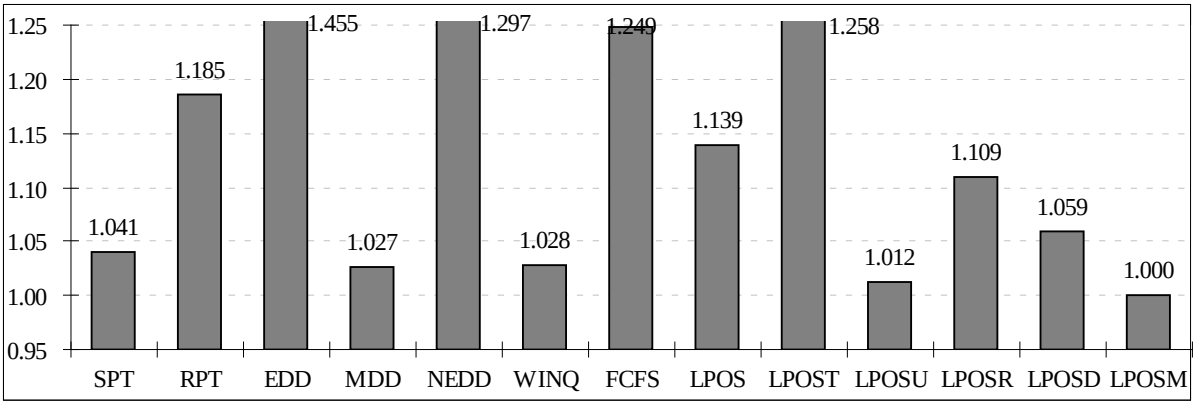


Figure VII.5. Tardiness STD in the busy system case.

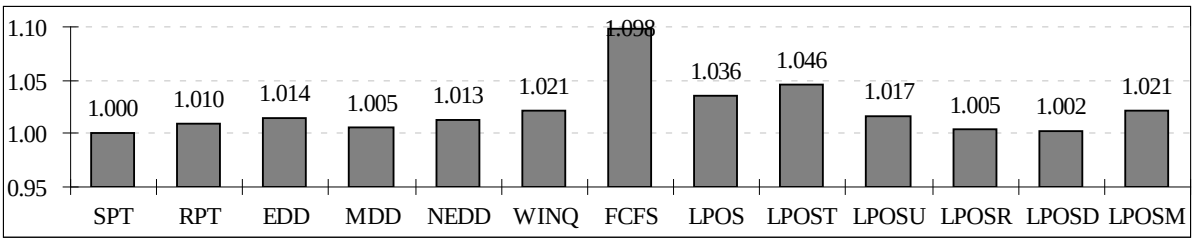


Figure VII.6. Percentage of tardy jobs in the busy system case.

The percentage of tardy jobs shows different results with flow time or other tardiness measures. Figure VII.6 represents that the performance of almost all dispatching rules are similar except for the FCFS rule. The SPT rule shows the best performance, and LPOSD, LPOSR, MDD, and RPT show only about 1% difference. Other rules also show under about 2% difference except for LPOS and LPOST which show about 4% difference.

2.3.2. Idle system

The LPOSR rule is generally the best in the idle system, and LPOST and LPOS follow it. The LPOS combined dispatching rules show good performance over the whole measures except for the percentage of tardy jobs which shows a different result with other measures.

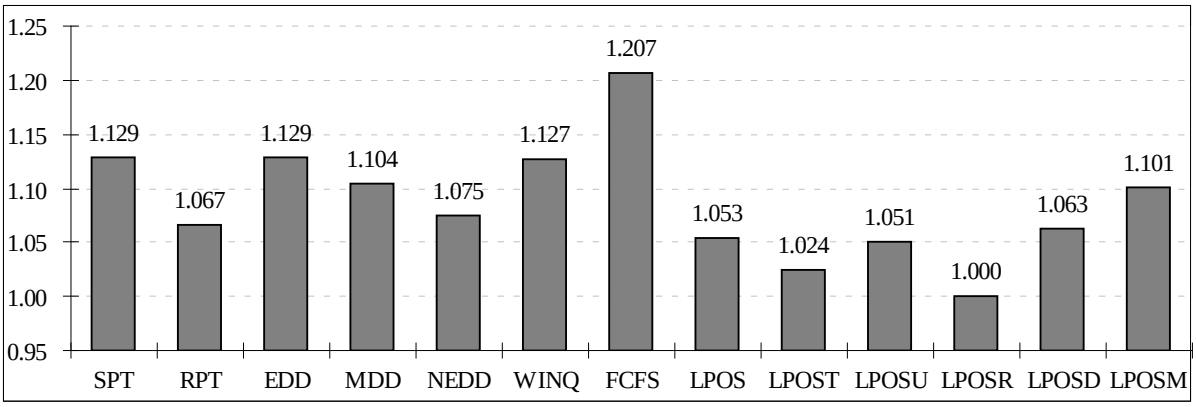


Figure VII.7. Mean flow time for the idle system case.

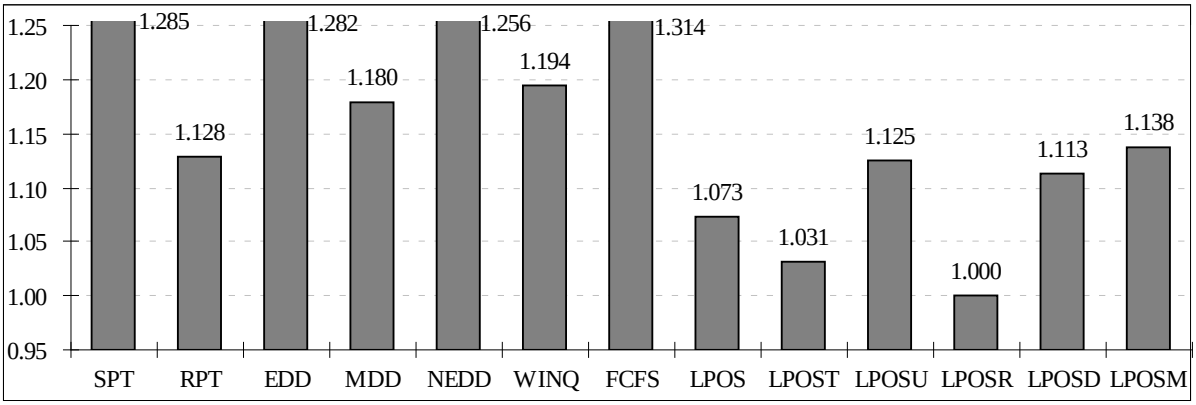


Figure VII.8. Flow time STD for the idle system case.

Figures VII.7 and VII.8 represent that LPOSR shows a quite better mean flow time and flow time STD than other rules; it shows over 5% better mean flow time than others except for LPOST, and over 10% better flow time STD than others except for LPOST and LPOS. The proposed dispatching rules are generally better than benchmark rules. RPT is the best among the benchmark rules but 6.7% and 12.8% worse than LPOSR in the mean and STD measures respectively. Although the mean flow time of LPOSM and MDD is similar as 1.101 and 1.104, the proposed rules are better than their parent dispatching rules in both measures; LPOST, LPOSR, LPOSD, and LPOSM respectively show 10.3% (= 1.129/1.024−1), 6.7%, 6.2%, and 0.3% better mean flow time than SPT, RPT, EDD, and MDD. They show over 10% better flow time STD except for LPOSM. Hence combining LPOS with existing dispatching rules can be considered to improve the performance by the flow time measures as expected in section VII.1.

Figures VII.9 and VII.10 represent that LPOSR also shows the best performance in the tardiness measures. It respectively shows 4.9% and 14.3% better mean tardiness and tardiness STD than RPT which shows the best performance among benchmark dispatching rules. LPOSR shows over about 5% better mean tardiness than all the other dispatching rules except for LPOST which is only 2.1% worse than LPOSR. The performance improvement effect by LPOS combining with benchmark dispatching rules is generally bigger than in the flow time measures. LPOST, LPOSR, LPOSD, and LPOSM respectively show 15.1% (= 1-1.052/1.314), 4.9%, 10.3%, and 5.6% better mean flow time than their mother dispatching rules.

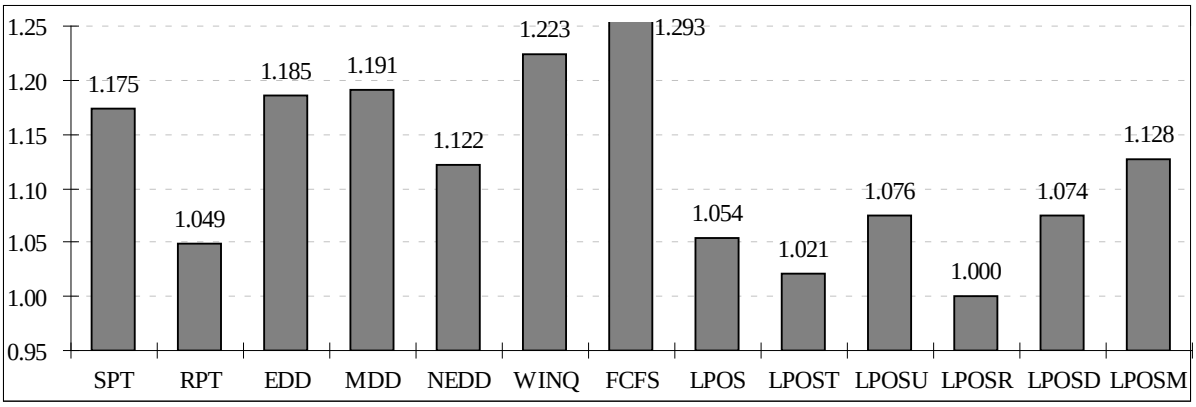


Figure VII.9. Mean tardiness for the idle system case.

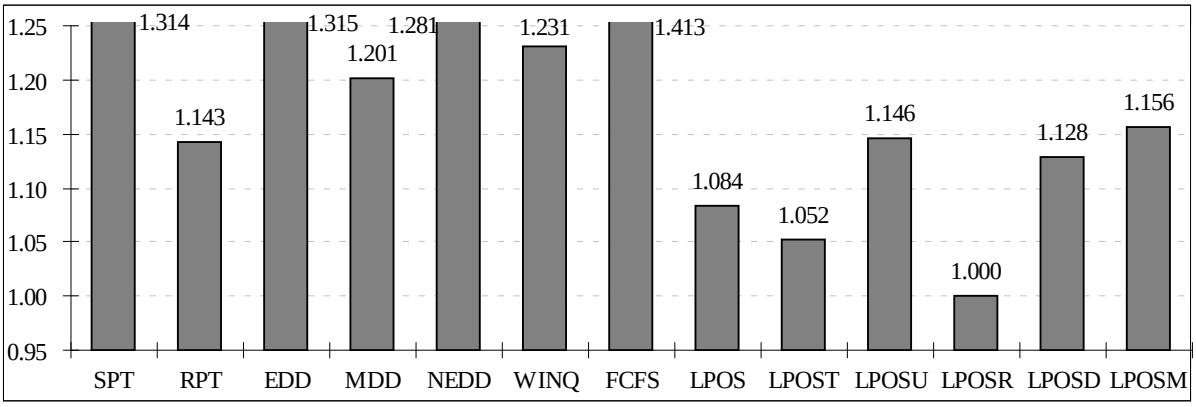


Figure VII.10. Tardiness STD for the idle system case.

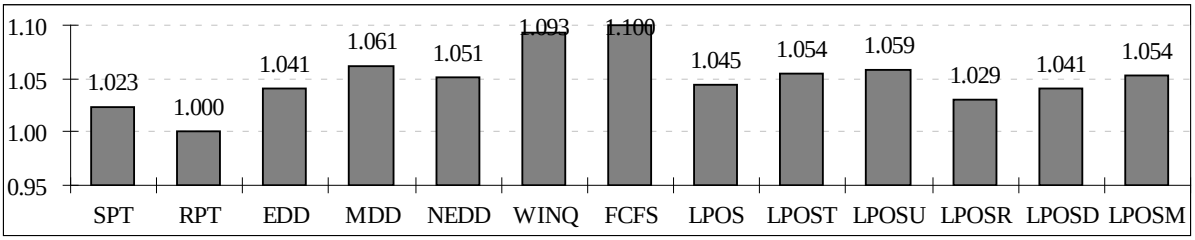


Figure VII.11. Percentage of tardy jobs for the idle system case.

The percentage of tardy jobs measure also shows different results with other measures like the busy system case, but their difference among dispatching rules is bigger (refer to figure VII.11). RPT shows over about 2% better performance than all the other rules, and FCFS still shows worst performance. Although the proposed dispatching rules were good in the mean tardiness and tardiness STD, they show bad performance in this measure; RPT shows 2.8% (= 1-1/1.029) better performance than LPOSR which is the best among the proposed dispatching rules.

2.4. Necessity of the dynamic dispatching rule allocation

Some of the proposed quality embedded dispatching rules showed better performances than conventional popular dispatching rules as discussed above, but we can find a clue to achieve better performances form the above simulation results.

Table VII.2. Performance difference between the two best dispatching rules for the busy and idle systems.

System State	Dispatching Rule	Flow time (minutes)		Tardiness (minutes)		
		Mean	STD	Mean	STD	PoTJ[a] (%)
Busy	MDD $^{(\alpha)}$	588.7	547.6	387.7	536.1	0.747
	LPOSU $^{(\beta)}$	574.3	539.6	367.7	528.3	0.756
	Improvement** (%)	2.45	1.46	5.15	1.44	-1.20
	P-value from t-test	0.356	0.452	0.315	0.454	0.316
Idle	RPT $^{(\gamma)}$	325.0	207.6	166.5	199.9	0.714
	LPOSR $^{(\delta)}$	304.7	184.1	158.7	175.0	0.735
	Improvement[b] (%)	6.26	11.33	4.67	12.50	-2.94
	P-value from t-test	0.101	0.052	0.260	0.050	0.066

[a] PoTJ: Percentage of tardy jobs.
[b] [improvement] = $(\alpha-\beta)/\alpha\times100$ or $(\gamma-\delta)/\gamma\times100$; the calculation results can be different with the value calculated with that in the table because of the value in the table is rounded from the original value.

The performance order among dispatching rules is different depending on the system states and target measure according to the simulation results in the above section VII.2.3. But the results do not definitely conclude the performance difference among dispatching rules, because the simulation results are not consistent even in the same condition; for example, the mean flow time of the best two dispatching rules LPOSU and MDD in the busy system case distribute from 474.3 to 715.5 and from 491.3 to 757.0 respectively. Table VII.2 shows the Student's t-test[1] results for the best two dispatching rules for each system states. The two-tailed t-test is done to verify superiority/inferiority among dispatching rules, and the unequal distribution is assumed because of the independency among simulations as the t-test in section VI.5.2. The LPOSU rule in the busy system is 2.45% and 5.15% better than MDD in the mean flow time and tardiness respectively, but LPOSU cannot be definitely considered to be superior to MDD because the p-values from t-tests show the superiority confidence level is only about 65%. The performance superiority in the idle system is more discriminative; the LPOSR rule's 6.26% and 4.67% superiority to RPT is about 90% and 75% confident respectively, but they are also under the 95% confidence level which is generally recognized as a significant difference in statistics.

Although the superiority of LPOSU and LPOSR cannot be definitely recognized from the perspective of the mean values of the flow time and tardiness, the LPOSR rule shows definite superiority than RPT in the idle system from the perspective of the STD. The standard deviation of the flow time and tardiness of LPOSR is respectively 11.33% and 12.50% better than RPT in the about 95% confidence level. Hence this thesis can conclude the variance decrease effect by quality information adoption to dispatching rules that is expected in section VII.1.

The simulation results represents the dispatching rule performance is different from system state to state; the best dispatching rule is different between the busy system and idle system, and the performance is distributed widely even in the same condition. Hence a better system performance is expected by monitoring the QRS state continuously and allocating different dispatching rules dynamically depending on the state. The dynamic dispatching rule allocation will be discussed in the next chapter VIII.

[1] This thesis utilized the Microsoft® office excel 2003 for all the mathematical and statistical calculations.

VIII. Real-time scheduling mechanism for the QRS

This chapter proposes a real-time scheduling mechanism on the multi-agent framework. The communication protocol among agents in the proposed framework in chapter V is discussed first. The protocol requires a system quality performance measure, hence a cost based measure is proposed with its detailed calculation method. The batch processing related topics are also discussed. The last section presents the knowledge-based dynamic dispatching rule allocation mechanism.

Abbreviated terms used in this chapter:

EDD	Earliest Due Date;
EWT	Expected operation Waiting Time;
FCFS	First Come First Serve;
LPOS	Lowest POS;
LPOSD	LPOS to Due date;
LPOSM	LPOS to Modified due date;
LPOSR	LPOS to Remaining total processing time;
LPOST	LPOS to operation processing Time;
LPOSU	LPOS to due date Urgency;
MDD	Modified Due Date;
NEDD	New EDD;
PDSP	used Product and Disassembled Subassembly/Part
QRS	Quality embedded Remanufacturing System;
RPT	Remaining Processing Time;
SPT	Shortest Processing Time;
STD	STandard Deviation;
TWK	Total Work content;
UML	Unified Modeling Language;
WINQ	Work In Next Queue.

1. Distinctive characteristics of the proposed scheduling mechanism

The proposed scheduling mechanism starts from the following basic idea:

"Each PDSP's competing with other PDSPs to maximize its performance is one of the solutions for system performance maximization,"

which is the basic concept of market-based negotiation mechanisms. The proposed mechanism grants each PDSP the decision right of selecting a process to follow among alternatives in order to make PDSPs compete without any central controls. As indirect mediators of the competition, resources' PDSP dispatching priorities are given by the allocated dispatching rules to pursue the whole system performance maximization. Hence the proposed mechanism can be classified as a hybrid negotiation mechanism (refer to section II.2.2).

The proposed mechanism has the following characteristics:

- Autonomous agents;
- Quality embedded bi-directional hybrid negotiation protocol;
- Dynamic dispatching rule allocation;
- Knowledge-based assistant.

PDSP and resource agents communicate with each other to achieve their assigned objectives, and they make decisions based on the information gathered from the communication; PDSPs try to maximize their own performance and resources try to maximize the whole system performance. Hence each agent is autonomous and their communication is hybrid. A resource dispatches PDSPs based on a dispatching rule announced by a knowledge agent, which dynamically allocates the best dispatching rules at the current system state to resources based on previously accumulated knowledge by off-line simulation with a heuristic approach. The following sections discuss the hybrid negotiation protocol and the knowledge-based dynamic dispatching rule allocation mechanism in detail.

2. Objective function

Applying a scheduling mechanism to the QRS is to obtain good performance, and a performance measure is required to judge if the performance is good or not. There are many kinds of performance measures; for example, tardiness, flow time, throughput, and so on. Although quality related features can be implicit in some performance measures, none of them targets to measure quality related effects. Therefore this research suggests a QRS performance measure which synthetically reflects quality related effects: the total remanufacturing cost. The performance measure for a remanufactured product is defined as follows (the subtitle indicating the remanufactured product is omitted for simplicity.[1]):

[1] For example, the due date of a used product ***rp*** should be expressed as t^d_{rp} (def. VII.1.7), but it is expressed as t^d without subscript ***rp***.

$$M^{perf} = \sum_{o \in PO} \sum_{r \in R_o} \left(c^{p}_{o,r} \cdot t^{p}_{o,r} \right) + c^{w} \cdot t^{w} + f^{odp} \left(Max[0, t^{c} - t^{d}] \right) + \sum_{p \in DP} c^{d}_{p} , \quad \text{(Def. VIII.2.1)}$$

where

$PO \subset \cup O_p$	set of processed operations for the remanufactured product ***rp*** remanufacturing;	(Def. VIII.2.2)
$R_o \subset R$	set of resources which processed the operation ***o*** during the ***rp*** remanufacturing;	(Def. VIII.2.3)
$DP \subset P$	set of disposed PDSPs during the remanufacturing;	(Def. VIII.2.4)
$c^{p}_{o,r} \in \mathbb{C}$	processing cost per unit time of the operation ***o*** by resource ***r***;	(Def. VIII.2.5)
$t^{p}_{o,r} \in \mathbb{T}$	processing time duration of the operation ***o*** by ***r***;	(Def. VIII.2.6)
$c^{w} \in \mathbb{C}$	waiting cost in a buffer per unit time;	(Def. VIII.2.7)
$t^{w} \in \mathbb{T}$	total waiting time in buffers during remanufacturing;	(Def. VIII.2.8)
f^{odp}	overdue penalty cost calculation function;	(Def. VIII.2.9)
$t^{c} \in \mathbb{T}$	remanufacturing completion time;	(Def. VIII.2.10)
t^{d} (t^{d}_{p})	due date (def. VII.1.7)	
$c^{d}_{p} \in \mathbb{C}$	disposal cost of a PDSP;	(Def. VIII.2.11)

where

O_p	set of operations (def. VI.2.25);
R	set of resources (def. VI.2.9);
P	set of PDSPs (def. VI.2.20).

The proposed QRS measure M^{perf} is applied to each remanufactured product, and the performance of the QRS is measured by an average, standard deviation (STD), or maximum value of all remanufactured products' M^{perf}.

The measure is a weighted sum of the processing cost, waiting cost, delay penalty cost, and disposal cost[1] of defect parts.[2] The quality in the QRS is handled in non-quantitative way, and forced definition of a synthetic quantitative quality measure is not preferable. Hence this research suggests a cost based measure in an indirect way. The formula encloses the overdue and waiting time effects as well as the PDSP disposal effects, because the cost increases due to overdue and longer processing time which can be caused by careful PDSP handling to decrease part defects or disposal. The proposed measure is cost based, hence a smaller value means a better performance.

3. Communication protocol

The agents in the proposed multi-agent system in chapter V pursue the minimization of the QRS performance measure M^{perf} through a clearly defined communication with other agents. The protocol is composed of two

[1] The disposal cost in this thesis also includes all the related substitution cost (refer to assumption I.2).
[2] During the execution or simulation of the QRS, each agent keeps its related time information and transmits them to the operation processing statistics information manager agent in the knowledge support agent group at the end of its life.

layers:

– layer 1: the message syntax and the passing method;
– layer 2: the communication sequence for the knowledge mounting[1], PDSP agent creation, and operation processing.

The agents communicate along the proposed sequence in the layer 2 to collect the required information for their decision making by sending/receiving messages which follow the syntax and the passing method defined in the layer 1.

3.1. Message syntax and message passing method (layer 1)

The messages among agents are string formatted and have the following syntax:

[Sender]:[Header]:[Detailed Information].

The message is partitioned by the character colon ':'. The [Header] specifies the message type which comprises two parts partitioned by the character slash '/': communication type and message objective. Zero or more occurrence of [Detailed Information] can be appended to the front two partitions depending on the message type. Those appended terms carry the main contents, therefore the agent which receives the message parses and converts the appended [Detailed Information] to a comprehensible format.

Table VIII.1 synthesizes the detail definition of the syntax. Each message communication type in the header has the following meanings:

AN ANnounce an agent's action or information to other agents;
RQ ReQuest other agents to return required information;
AW AnsWer to the requested information from other agents.

For example, the best dispatching rule announcement by the dispatching rule allocator agent can be DRA:AN/BDR:SPT, where DRA and SPT means the ID of the dispatching rule allocator agent and the selected best dispatching rule respectively.

The agent communications can be one of two types: one-directional and bidirectional. One-directional communication utilizes AN type messages to just announce some information to other agents, while bidirectional communication utilizes RQ and AW type messages to request some information and to answer the requests. The header's second part of the answer message is the same with that of the corresponding request message; for example, the header of the PDSP agent's estimated waiting time (EWT) request message to a resource agent is RQ/EWT, and the header of the answer message from the resource agent is AW/EWT.

[1] The knowledge mounting means loading the generated knowledge/scenario map on the memory space of the dispatching rule allocator agent. It is completed when all the required information is loaded and ready to be utilized by the dispatching rule allocator agent (refer to figure VIII.6).

Table VIII.1. Message syntax for the communication protocol.

Header symbol		Detailed Information	Sender agent[b]	Receiver agent[b]	Objective of messages
Type	Objective[a]				
AN	NK	Knowledge	SA	KA	New Knowledge
AN	PD	PDSP information	PA	LA	PDSP Disassembly
AN	PR	PDSP information	PA	LA	PDSP Reassembly
AN	TP	-	RA	PA	Turn to be Processed
AN	QES	-	WA	PA	Quality Examination Start
AN	QER	Examined quality	WA	PA	Quality Examination Result
AN	BDR	Dispatching rule ID	KA	WA	Best Dispatching Rule
AN	TAL	-	ANY	ANY	Terminate Agent Life
RQ	HPI	-	SA	KA	Historical Performance Information
RQ	EWT	PDSP information	PA	RA	Estimated Waiting Time
RQ	ROP	PDSP information	PA	RA	Registration of Operation Processing
RQ	CR	-	PA	RA	Cancelation of Registration
RQ	QE	PDSP ID	RA	WA	Quality Examination
AW	CMR	-	ANY	ANY	Confirmation of Message Receive
AW	CAH	-	ANY	ANY	Completion of Announced message Handling
AW	*	Requested information	ANY	ANY	Answer to the requested information

[a] Abbreviation of objective of messages;
[b] KA: knowledge agent, SA: simulator agent, WA: workstation agent, RA: resource agent, BA: buffer agent, PA: PDSP agent, ANY: any agent.

The agents in the proposed multi-agent structure communicate with each other with a push message passing method, because all communications are carried on inside of the virtual system. Hence an agent sends a message directly to the receiver agent instead of broadcasting it to all the agents. The message receiver agent should always return a confirmation message to the sender agent that the message is correctly delivered; this message receiving confirmation process is mandatory to avoid deadlock from non-delivered messages. The sender agent checks the receiver agent's availability when it does not receive a confirmation message within a certain time after its message sending, and the sender agent should find other alternatives in case the receiver agent is unavailable.

3.2. Communication sequence (layer 2)

This section represents the detailed communication sequence with modified unified modeling language (UML) sequence diagrams (refer to figure VIII.1 - VIII.4), where belonging states are indicated on the agent life line (dotted vertical line). The meaning of the marked state numbers can be found in the state transition diagram of each agent in section V.2.1. The states during the communication are omitted in case the communication does not affect the agent state or an agent has only one state. The numbers enclosed by the parenthesis of the brief sequence description in the three subsections below indicate the corresponding step numbers in the sequence diagrams.

3.2.1. Communication for the knowledge mounting

The knowledge on the best dispatching rules for resources depending on the system states is the fundamental

information of the proposed scheduling mechanism. The simulator agent generates the knowledge with a heuristic method and mounts it on the dispatching rule allocator agent. Section VIII.6 will discuss the knowledge and the knowledge generation heuristic in detail. Figure VIII.1 represents the detailed communication sequence of a knowledge update which can be abridged as follows:

a. The simulator agent gathers the required information (1 - 3);
b. The simulator agent runs the simulation (4);
c. The simulator agent synthesizes the simulation results and generate knowledge (5);
d. The simulator agent mounts the knowledge to the dispatching rule allocator agent (6 - 8).

The knowledge is created and updated on demand by the person in charge of the real-world QRS control (refer to section V.2.1.4), hence the QRS controller should invoke the simulator agent to start updating process for the knowledge refinement.

3.2.2. Communication for the PDSP creation

New PDSP agents are created in the following two cases: new used product arrival and PDSP disassembly/reassembly.

The communication for the used product arrival case is simple. Used product arrivals are recognized by a used product arrival detector in the real-world and announced to the used product arrival manager agent. The used product arrival manager agent creates a corresponding PDSP agent, synchronizes the created agent with the corresponding used product in the real-world, and starts the created agent's life. The required information is passed at the creation time: the alternative processes, facilities layout, due date, element ID of the used product in the real-world. Figure VIII.2 shows the detailed communication sequence.

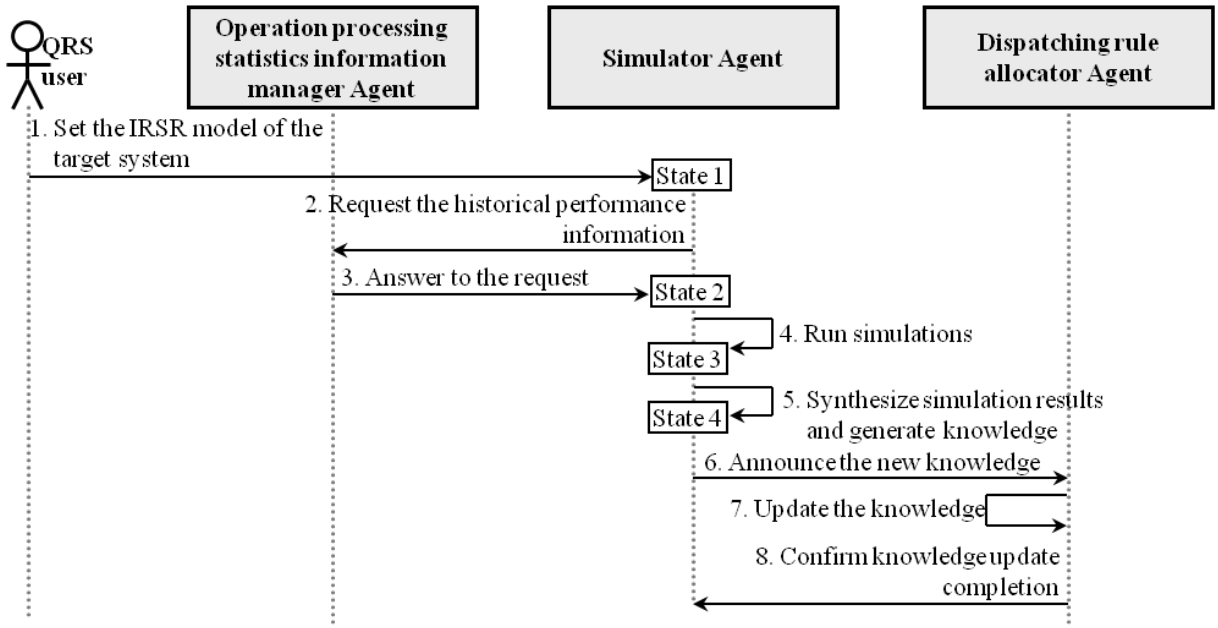


Figure VIII.1. Communication sequence diagram of the knowledge mounting.

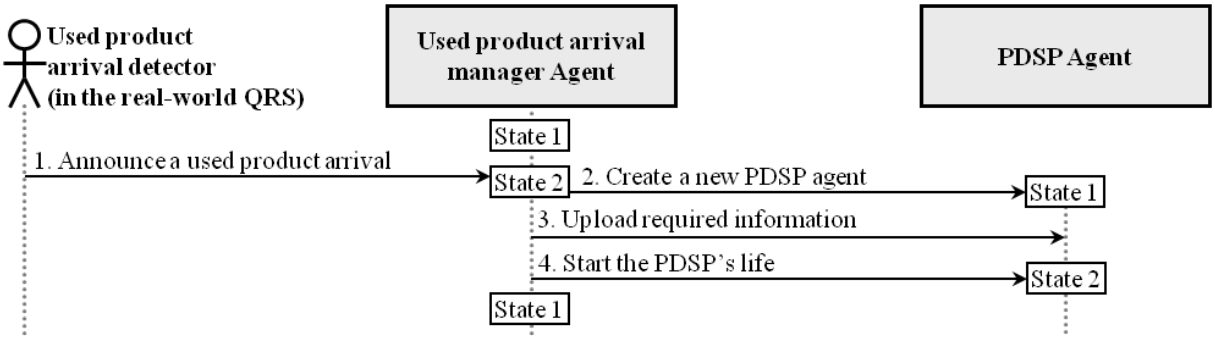


Figure VIII.2. Communication sequence diagram of the PDSP creation by the used product arrival.

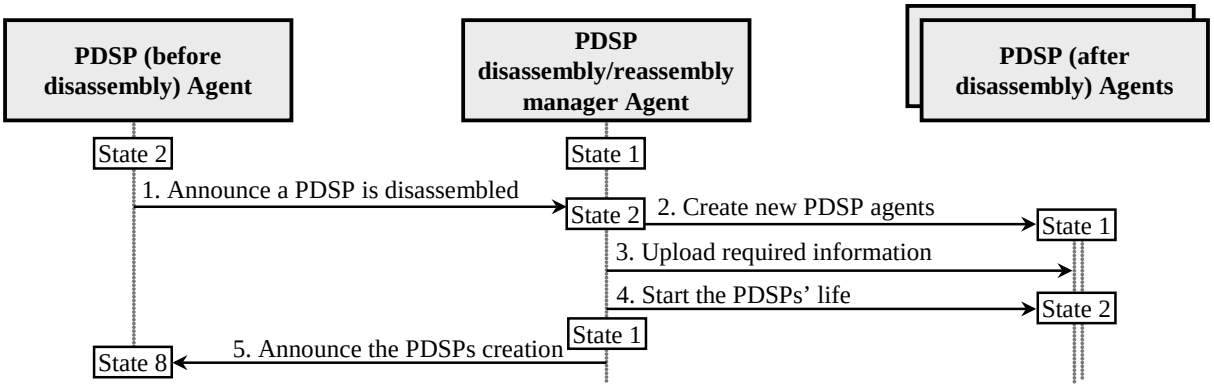


Figure VIII.3. Communication sequence diagram of the PDSP disassembly.

The communication for the PDSP disassembly or reassembly case is similar with that for the used product arrival case. Figure VIII.3 represents the communication sequence for the case in detail. The quality examination result after disassembly is announced to the PDSP agent which corresponds to the PDSP before disassembly, and it is passed to the PDSP disassembly/reassembly manager agent in the disassembly completion cases (refer to figure IV.2), because the disassembled PDSPs require new PDSP agents. The disassembled PDSPs should inherit some information from the PDSP before disassembly: the quality, due date, entrance time to the system, processing and waiting time of operations up to the disassembly completion time. Hence the PDSP disassembly/reassembly manager agent receives that information to upload it into the created new PDSP agents. The new agent creation completion is announced to the PDSP agent corresponding to the PDSP before disassembly to end its life. The sequence diagram and explanation on the reassembly case is omitted due to its similarity; there is only one difference that the PDSP after reassembly is single and PDSPs before reassembly are two or more.

The brief communication sequence of both creation cases are as follows:

a. The PDSP life management agent[1] receives the messages on new PDSP appearances (1);
b. The PDSP life management agent creates PDSPs and upload required information (2 and 3);
c. The PDSP life management agent starts the created PDSPs' lives (4 and 5).

[1] The PDSP life management agent indicates either the used product arrival manager agent (figure VIII.2) or the PDSP disassembly/reassembly manager agent (figure VIII.3) in the PDSP life management group (refer to figure V.4).

3.2.3. Communication for the operation processing

The PDSP, resource, and workstation agents are directly involved in the communication to process each remanufacturing operation. The objective of the three agents involved in the operation processing is to complete the used product remanufacturing from the perspective of the whole system performance maximization, and the objective is distributed to the two main agents as follows:

– PDSP agent: try to complete its process with its maximum performance without considering other agents;
– Resource agent: do requested operations with the sequence ordered by an allocated dispatching rule.

The workstation agents have no objective; they have no decision making functions but gather information from the corresponding real-world element and pass it to other agents.

The communication protocol for the operation processing is PDSP oriented as explained in section V.2.1.1; the remanufacturing process is led by the PDSP agents. The remanufacturing process is completed by each PDSP's completing its operations one by one, hence this section represents the communication sequence only for one operation completion. The iterative communication following the sequence will make a used product be remanufactured. Figure VIII.4 represents the detailed communication sequence of one operation processing which can be summarized as follows:

a. The PDSP agent collects the EWT information of each operation in the alternative processes and selects an operation to do and a resource to process the selected operation (1 - 5);
b. The PDSP agent requests an operation processing to the selected resource agent and waits its turn (6 - 10);
c. (Optional) The PDSP agent goes back to the first step to reselect an operation and a resource in case its waiting time exceeds a certain tolerance, (11 – 13);
d. The resource agent selects a PDSP to process and waits for its arrival (14 - 18);
e. The resource processes operations of selected PDSPs and requests the belonged workstation agent to examine the output PDSP quality (19 and 20);
f. The workstation examines the PDSP quality, and the result is announced to the PDSP agent (21 - 23);
g. The PDSP agent updates its information according to the operation processing results and decides the next operation to process (24 and 25).

The steps a and c are needless when the PDSP has no alternative operations and resources. While one PDSP is selected to be processed by resources (step 15), batch resources can select one or more PDSPs as the same time and send a message to all the selected PDSPs (step 16). The EWT (step 1 - 4) means the waiting time from the operation registration to the resource (step 5) until getting a message to come to the resource for operation processing (step 16); the PDSP transportation time is not counted, because the time is assumed to be zero in this thesis (refer to assumption I.9). A PDSP may need to wait its quality examination result after its operation end in a resource (step 20 and 21) when the workstation is examining the other PDSPs' quality. But the waiting time for the quality examination is not considered either, because this thesis also assumes the quality examination time is comparably small in comparison with the operation processing time (refer to assumption I.12).

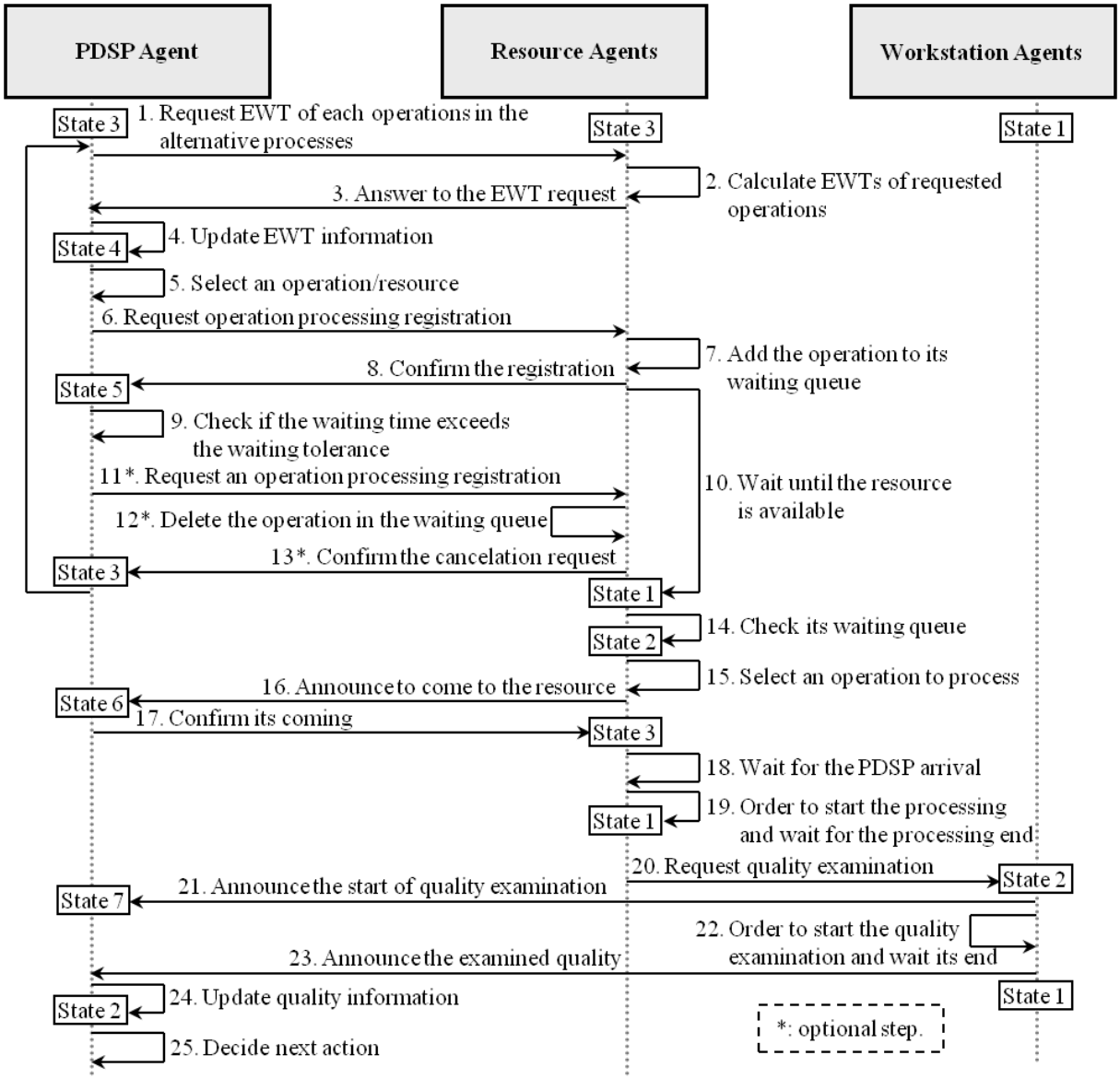


Figure VIII.4. Communication sequence diagram of the operation processing.

The PDSPs and resources in the real-world remanufacturing system act in according to the corresponding agents' execution command and send their state to the agents. A PDSP agent is invoked to start the information collection (step 1) by the signal of the corresponding real-world PDSP's arrival at a buffer. The PDSP agent's receiving the processing turn message (step 16) make the real-world PDSP transfer to the resource. The PDSP's arrival at the resource is detected by the real-world resource and the processing start is invoked (step 19). The real-world resource processes the operation and passes the processed PDSPs to the quality examination machine or to the person in charge of the examination. Those actions are announced to the corresponding resource agent which requests the quality examination start (step 20). The quality examination result is updated to the workstation agent and transferred to the PDSP agent (step 23). Last, the PDSP agent decides the next action to do and commands the corresponding real-world PDSP to transfer to an appropriate buffer.

This thesis involves the resource quality in the PDSP's operation/resource selection (step 5) which will be

discussed in section VIII.4. It is preferable for the up-to-date information that the PDSP agent requests the quality of resources to the operation processing statistics information manager agent in real-time. But the information is once mounted on the PDSP agent by the product arrival manager agent at the creation time, because the resource quality changes little during the PDSP's life time. This approach can reduce communications during the PDSPs operation/resource selection. The used product arrival manager agent maintains the up-to-date resource quality information by periodic communications with the operation processing statistics information manager agent. While the PDSP agent requires the resource quality information which is comparably stable, the resource agent requires real-time information of the PDSPs which requested operation processing. Hence the PDSP agents send all their information to the resources at the requesting time; for example, the current quality, due date, waiting time up to the requesting time, and so on. The PDSP processing sequence by resources is decided based on this information by applying the allocated dispatching rules.

The following section VIII.4 discusses the detailed sub-problems related to the operation processing; the EWT calculation by the resource agent (step 2), the operation/resource selection (step 5), and waiting time tolerance calculation (step 9) by the PDSP agent.

4. Operation and resource selection by PDSPs

4.1. Estimated waiting time (EWT) calculation

The PDSP agent requests EWT information which is used for the operation/resource selection to resource agents. The resource agents return corresponding values calculated based on historical waiting time information in similar conditions which is classified by the PDSP's requesting operation and allocated dispatching rule at the operation requesting time. This approach is motivated from the following two ideas;

– the PDSP dispatching priorities may be similar if its requesting operation and the applied dispatching rule are the same;
– the more PDSPs are in the resource's waiting queue, the longer time the PDSP waits for.

Hence each agent corresponding to a resource $\boldsymbol{r}$ accumulates the historical information of the dispatched PDSPs to the information set $\boldsymbol{S}^{\boldsymbol{HI}}_{\boldsymbol{r}}$ as follows:

$\boldsymbol{S}^{\boldsymbol{HI}}_{\boldsymbol{r}} = \{(\boldsymbol{o}, \boldsymbol{dr}^{\boldsymbol{rt}}, \boldsymbol{n}^{\boldsymbol{wq}}, \boldsymbol{t}^{\boldsymbol{rw}}_{\boldsymbol{o}}) \mid \boldsymbol{n}^{\boldsymbol{wq}} \in \mathbb{I}^{+}, \boldsymbol{dr}^{\boldsymbol{rt}} \in \boldsymbol{DR}^{\boldsymbol{A}}, \boldsymbol{o} \in \cup \boldsymbol{O}_{\boldsymbol{p}}, \text{ and } \boldsymbol{t}^{\boldsymbol{rw}}_{\boldsymbol{o}} \in \mathbb{T}\}$

set of historical information on realized waiting time by a resource $\boldsymbol{r}$ in case a PDSP requested an operation $\boldsymbol{o}$ and the resource applied a dispatching rule $\boldsymbol{dr}^{\boldsymbol{rt}}$, (Def. VIII.4.1)

where

$\boldsymbol{n}^{\boldsymbol{wq}}$ number of PDSPs in the resource's waiting queue at the PDSP's operation processing request time; (Def. VIII.4.2)

t^{rw}_o realized waiting time of the PDSP which is processed by r; [1] (Def. VIII.4.3)

DR^A set of available dispatching rules; (Def. VIII.4.4)

O_p set of operations (def. VI.2.25);

Resource agents return the following values based on the historical record set S^{HI}_r to the EWT requesting PDSP agent:

$$t^{ew}_o = \underset{(o,dr^{rt},n^{wq},t^{rw}_o)\in S^{HI}_{r,ro,cdr}}{avg}\left(t^{rw}_o / n^{wq}\right)\times n^{wq,current}$$ EWT of the PDSP requesting operation o processing; (Def. VIII.4.5)

$$v^{ewt}_o = \underset{(o,dr^{rt},n^{wq},t^{rw}_o)\in S^{HI}_{r,ro,cdr}}{var}\left(t^{rw}_o / n^{wq}\right)\times n^{wq,current}$$ variance of t^{rw}_o, (Def. VIII.4.6)

where

$S^{HI}_{r,ro,cdr} = \{(ro, cdr, n^{wq}, t^{rw}_o)\} \subset S^{HI}_r$

subset of S^{HI}_r; its instances are the base information for the EWT calculation of the PDSP requesting operation $ro \in \cup O_p$, when a resource applies dispatching rule $cdr \in DR^A$;

$n^{wq,current}$ the number of PDSPs in the waiting queue at the current time.

The returned EWT values to the PDSP are used for the calculation of not only the estimated total cost based performance $M^{m\text{-}perf}$ which will be defined in the next section VIII.4.2 but also the waiting tolerance which will be discussed in section VIII.4.4.

4.2. Operation selection

The PDSP agent should select an operation to process when it has two or more alternative operations, and the PDSP agent in the proposed scheduling mechanism selects it by comparing the estimated performance of each operation selection case. The estimated performance of a PDSP means the expected performance of a to-be-remanufactured product with the PDSP at the remanufacturing completion time. The estimation requires the EWT of all the following operations and quality of resources in charge of the operations. The operation processing communication protocol contains the communication between PDSP and resource agents for the PDSP's EWT information collection (refer to step 1 - 3 in figure VIII.4). But no communication is necessary for the resource quality information collection, because the PDSP maintains the information from the PDSP creation time (refer to section VIII.3.2.3).

The objective of estimating the performance of each alternative operation selection case is to compare the performance of the case with the other selection cases. Hence the calculation only on the upcoming process is enough, because the performance of the already completed process is the same to all alternative operations. The proposed cost based performance measure M^{perf} (def. VIII.2.1) calculation for the upcoming process requires

[1] Realized waiting time is from the operation request time to the resource r (right after step 6) to the operation processing start time by the resource (right before step 19) (refer to figure VIII.4).

future information which cannot be gathered at the calculation time: the waiting/processing time of operations to be processed, the parts to be disposed of during the upcoming process, and the time of remanufacturing to be completed. This thesis estimates future information with a probabilistic and stochastic approach by calculating $\boldsymbol{M}^{m\text{-}perf}$ which is a modified version of $\boldsymbol{M}^{perf}$. The $\boldsymbol{M}^{m\text{-}perf}$ is defined as follows:

$$\boldsymbol{M}^{m-perf} = \sum_{o \in PO} \left(\boldsymbol{c}^{p}{}_{o} \cdot \boldsymbol{t}^{ep}{}_{o} \cdot \boldsymbol{p}^{os}{}_{o} \right) + \boldsymbol{c}^{w} \cdot \boldsymbol{t}^{ew} + \sum_{p \in DP} \boldsymbol{c}^{d}{}_{p} \cdot \boldsymbol{p}^{d}{}_{p} + \boldsymbol{f}^{odp} \left(\boldsymbol{max}(0, \boldsymbol{t}^{ec} - \boldsymbol{t}^{d}) \right), \quad \text{(Def. VIII.4.7)}$$

where the additional or modified terms to $\boldsymbol{M}^{perf}$ are defined as follows:

$\boldsymbol{c}^{p}{}_{o} \in \mathbb{C}$	average processing cost per unit time of an operation $\boldsymbol{o}$ by resources in charge;	(Def. VIII.4.8)
$\boldsymbol{t}^{ep}{}_{o} \in \mathbb{T}$	estimated processing time of $\boldsymbol{o}$ in case it is to be processed;	(Def. VIII.4.9)
$\boldsymbol{p}^{os}{}_{o} \in \mathbb{R}^{0+} \leq 1$	probability of $\boldsymbol{o}$ to be processed;	(Def. VIII.4.10)
$\boldsymbol{c}^{w}$	waiting cost per unit time (def. VIII.2.7);	
$\boldsymbol{t}^{ew} \in \mathbb{T}$	estimated total waiting time during the upcoming process;	(Def. VIII.4.11)
$\boldsymbol{c}^{d}{}_{p}$	disposal cost of a PDSP $\boldsymbol{p}$ (def. VIII.2.11);	
$\boldsymbol{p}^{d}{}_{p} \in \mathbb{R}^{0+} \leq 1$	probability of $\boldsymbol{p}$ to be disposed of during the upcoming process;	(Def. VIII.4.12)
$\boldsymbol{f}^{odp}$	overdue penalty cost calculation function (def. VII.2.9);	
$\boldsymbol{t}^{ec} \in \mathbb{T}$	estimated remanufacturing completion time;	(Def. VIII.4.13)
$\boldsymbol{t}^{d}$ ($\boldsymbol{t}^{d}{}_{p}$)	due date (def. VII.1.7).	

Although the estimated performance $\boldsymbol{M}^{m\text{-}perf}$ calculation requires each operation's processing probability $\boldsymbol{p}^{os}{}_{o}$, the possible combinations of operation selections and PDSP qualities after the operation execution for the upcoming process are too many to calculate $\boldsymbol{M}^{m\text{-}perf}$ directly. The more the number of possible alternative upcoming processes are, the more complex calculation is required. Hence this thesis approaches this problem with recursive calculations.

Figure VIII.5 represents the structure of recursive function calls for the $\boldsymbol{M}^{m\text{-}perf}$ calculation. Hence $\boldsymbol{M}^{m\text{-}perf}$ for a candidate operation $\boldsymbol{o} \in \boldsymbol{O}_{p}$ (def. VI.2.25) of a PDSP $\boldsymbol{p}$ with input quality $\boldsymbol{iq} \in \boldsymbol{Q}_{p}$ (def. IV.2.1) can be calculated as follows:

$$\boldsymbol{M}^{m-perf} = \boldsymbol{f}^{epdc-r}(\boldsymbol{p}, \boldsymbol{o}, \boldsymbol{iq}, \boldsymbol{t}^{current}) + \boldsymbol{f}^{odp} \left(\boldsymbol{max}(0 \;\; \boldsymbol{f}^{ect-r}(\boldsymbol{p}, \boldsymbol{o}, \boldsymbol{iq}, \boldsymbol{t}^{current}) - \boldsymbol{t}^{d}) \right)$$

where

$\boldsymbol{f}^{edpc\text{-}r}$	operation processing and PDSP disposal cost estimation function in the recursive way;	(Def. VIII.4.14)
$\boldsymbol{t}^{current}$	current time; time instance of the calculation start time (def. VII.1.8);	
$\boldsymbol{f}^{odp}$	overdue penalty cost calculation function (def. VIII.2.9);	
$\boldsymbol{f}^{ect\text{-}r}$	remanufacturing completion time estimation function in the recursive way;	(Def. VIII.4.15)
$\boldsymbol{t}^{d}$	due date (def. VII.1.7).	

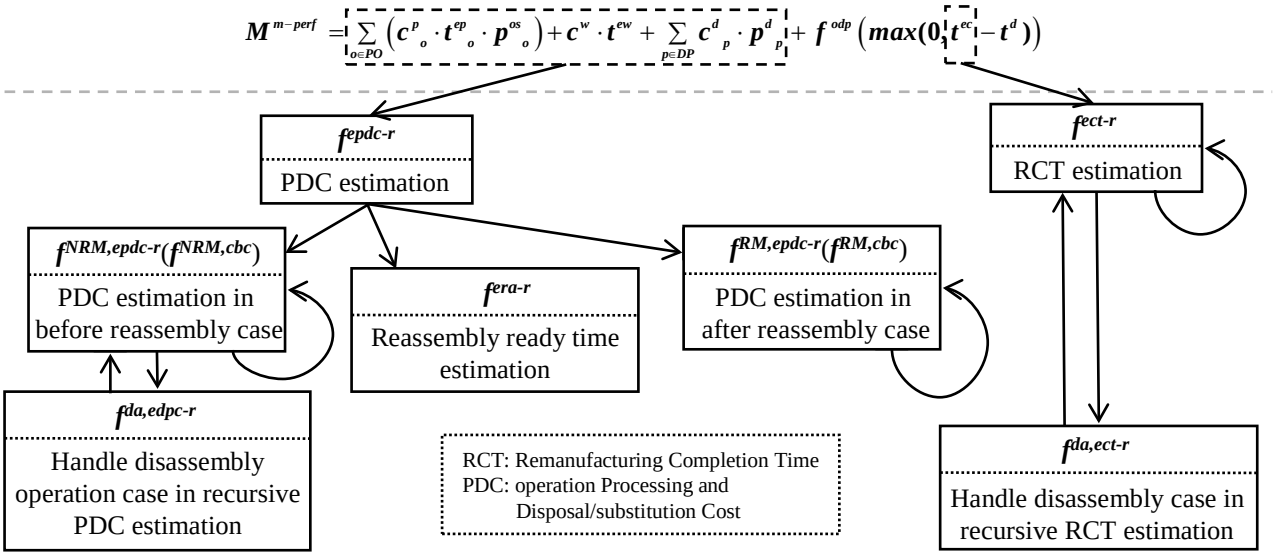


Figure VIII.5. Structure of estimating the cost based performance measure by recursive function calls.

Consecutive subsections explain the details of the above functions. The delay penalty cost estimation is discussed first than other functions for the explanation efficiency.

4.2.1. Delay penalty cost estimation

The estimated completion time t^{ec} in the delay penalty term (last term) of $M^{m\text{-}perf}$ can be calculated by the following estimated remanufacturing completion time calculation function $f^{ect\text{-}r}$ (def. VIII.4.15) (refer to figure VIII.5):

$$f^{ect-r}(p,o,iq,t^{ps}) = \sum_{(oq,rto_{oq},n^{rw},n^{rw}) \in OQ_{o,iq}} \left(rto_{oq} \times \begin{pmatrix} t^{ec}{}_{o,iq,oq} (= t^{ps} + t^{ew}{}_o + t^{ep}{}_{o,iq,oq} \times n^{rw}) & , if\ o = o^e{}_{rap_p} \\ f^{ect-r}(p, o^e{}_p, nd_p, t^{ec}{}_{o,iq,oq}) & , else\ if\ oq \in QS^D{}_p \\ f^{da,ect-r}(p,o,oq,t^{ec}{}_{o,iq,oq}) & , else\ if\ o = o^e{}_p \\ \underset{no \in \lambda^{fno}{}_{p,oq}(o)}{avg} f^{ect-r}(p,no,oq,t^{ec}{}_{o,iq,oq}) & , otherwise \end{pmatrix} \right),$$

of which each of the case by case function is for follows:

- the operation o is the end operation of the whole remanufacturing process;
- the output quality of the PDSP p after o is the quality to be disposed of;
- o is a disassembly or reassembly operation;
- other cases,

and the sub-function for disassembly/reassembly case of which is defined as follows:

$$f^{da,ect-r}(p,o,o\ qt^{ec}{}_{o,iq,oq}) = \underset{dp \in \lambda^d(o)}{max} \begin{pmatrix} f^{ect-r}(dp, o^e{}_{dp}, nd_{dp}, t^{ec}{}_{o,iq,oq}) & , if\ \lambda^{cq}(oq,dp) \in QS^D{}_{dp} \\ f^{ect-r}(dp, o^s{}_{dp}, \lambda^{cq}(oq,dp), t^{ec}{}_{o,iq,oq}) & , otherwise \end{pmatrix}, \quad \text{(Def. VIII.4.16)}$$

of which each of the case by case function is for follows:

– the output quality of the PDSP $\boldsymbol{p}$ after the operation $\boldsymbol{o}$ is the quality to be disposed of;
– other cases,

where

$\boldsymbol{t^{ps}} \in \mathbb{T}$	possible start time of the operation;	(Def. VIII.4.17)
$\boldsymbol{OQ_{o,iq}}$	set of possible output qualities $\boldsymbol{oq}$, their frequency ratio $\boldsymbol{rto_{oq,n^{rw}}}$, and the number of operation rework times $\boldsymbol{n^{rw}}$ including the first trial, for $\boldsymbol{o}$ and input quality $\boldsymbol{iq}$ (refer to appendix H for the detailed calculation of the set);	(Def. VIII.4.18)
$\boldsymbol{t^{ec}_{o,iq,oq}} \in \mathbb{T}$	estimated completion time of $\boldsymbol{o}$ for $\boldsymbol{iq}$ and $\boldsymbol{oq}$;	(Def. VIII.4.19)
$\boldsymbol{t^{ew}_{o}}$	estimated waiting time (def. VIII.4.5);	
$\boldsymbol{t^{ep}_{o,iq,oq}} \in \mathbb{T}$	estimated processing time of $\boldsymbol{o}$ for $\boldsymbol{iq}$ and $\boldsymbol{oq}$;	(Def. VIII.4.20)
$\boldsymbol{o^{e}_{rap_p}}$, $\boldsymbol{o^{e}_{dp}}$ ($\boldsymbol{o^{e}_{p}}$)	end operation of $\boldsymbol{rap_p}$ or $\boldsymbol{dp}$ (def. VI.2.27);[1]	
$\boldsymbol{rap_p} \in \boldsymbol{P^R}$	remanufactured product with $\boldsymbol{p}$;	(Def. VIII.4.21)
$\boldsymbol{dp} \in \boldsymbol{P}$	disassembled PDSP from $\boldsymbol{p}$;	
$\boldsymbol{nd_p} \in \boldsymbol{Q_p}$	no defect quality of $\boldsymbol{p}$;	(Def. VIII.4.22)
$\boldsymbol{QS^{D}_{p}} \subset \boldsymbol{Q_p}$	set of $\boldsymbol{p}$'s qualities to be disposed of;	(Def. VIII.4.23)
$\boldsymbol{\lambda^{fno}_{p,oq}}$: $\boldsymbol{Q_p} \rightarrow 2^{Op}$	function specifying $\boldsymbol{p}$'s filtered next operations by its $\boldsymbol{oq}$;	(Def. VIII.4.24)
$\boldsymbol{\lambda^{d}}$	disassembly/reassembly relationship function (def. VI.2.23);	
$\boldsymbol{\lambda^{cq}}$	quality mapping function among PDSPs (def. VI.2.24);	
$\boldsymbol{o^{s}_{dp}}$ ($\boldsymbol{o^{s}_{p}}$)	start operation of $\boldsymbol{dp}$ (def. VI.2.26),	

where

$\boldsymbol{P^R}$	set of remanufactured products (def. VI.2.22);
$\boldsymbol{Q_p}$	set of possible qualities (def. IV.2.1);
$\boldsymbol{P}$	set of PDSPs (def. VI.2.20).

Here the $\boldsymbol{f^{da,ect\text{-}r}}$ is the disassembled/reassembled PDSPs' estimated completion time merging function which will be explained later. The filtered next operation specification function $\boldsymbol{\lambda^{fno}_{p,oq}}$ is defined because next operations of an operation $\boldsymbol{o}$ depend on the quality of $\boldsymbol{p}$ after the operation. The estimated operation completion time $\boldsymbol{t^{ec}_{o,iq,oq}}$ can be easily gathered by summing the possible starting time $\boldsymbol{t^{ps}}$, estimated waiting time $\boldsymbol{t^{ew}_{o}}$, and estimated processing time $\boldsymbol{t^{ep}_{o,iq,oq}}$.

[1] The end operation of a reassembled product $\boldsymbol{rap_p}$ means the end operation after all post-reassembly processing is completed; for example, a reassembled used product may be painted, tested, and so on. The end operation of a disassembled part $\boldsymbol{dp}$ corresponds to its reassembly operation.

$f^{ect\text{-}r}$ should be handled in a different way depending on the output quality and the operation type. The first condition is for the case the recursive function call reaches the end of the whole remanufacturing process; the operation completion time is just returned in the case. The second condition is for the case the output quality is the quality to be disposed of. The disposed PDSP is always substituted with a new one which is ready to be reassembled. The recursive function $f^{ect\text{-}r}$ is recalled with parameters of the last operation of the PDSP p, the no defect quality nd_p of the substituted PDSP, and the end time of the operation o. The third condition is for the case o is the last operation of p which is either a disassembly or reassembly operation because of filtering the remanufactured product's last operation by the first condition, and the disassembled PDSPs' estimated completion time merging function $f^{da,ect\text{-}r}$ handles it. This thesis models the disassembly/reassembly relationship among PDSPs using the same function λ^d, hence the reassembly case can also be calculated by $f^{da,ect\text{-}r}$. The last condition is for the general condition; the average value of the estimated remanufacturing completion time t^{ec} for alternative next operation selection cases is returned.

The disassembly case handling function $f^{da,ect\text{-}r}$ in $f^{ect\text{-}r}$ is used to get the maximum value of the disassembled PDSPs completion time. Each disassembled PDSP right after disassembly operation can have two conditions; the first and second conditions are for the case of to be disposed of and to be processed. The PDSP to be disposed of right after disassembly should be handled in the same way with the second condition of $f^{ect\text{-}r}$; the PDSP is considered as ready to be reassembled, because it is directly substituted with a new one without any further refurbishment. The case of to be processed PDSP calls the function $f^{ect\text{-}r}$ with the disassembled PDSP dp's start operation o^s_{dp}.

4.2.2. Processing and disposal cost estimation

The processing and disposal cost calculation terms of $M^{m\text{-}perf}$ (def. VIII.4.7) can be calculated by the processing and disposal cost estimation function $f^{epdc\text{-}r}$ (def. VIII.4.14) (refer to figure VIII.5) which is defined as follows:

$$f^{epdc-r}(p,o,iq,t^{ps}) = \begin{cases} f^{RM,epdc-r}(p,o,iq,t^{ps}) & ,if\ \ p \in P^R \\ f^{NRM,ect-r}(p,o,iq,t^{ps}) + c^{p}_{rao_p} \cdot t^{ep}_{rao_p,nd_{rap_p},nq_{rap_p}} & \\ \qquad + f^{RM,epdc-r}(rap_p, o^s_{rap_p}, nd_{rap_p}, t^{era}) & ,otherwise \end{cases},$$

of which each of the case by case function is for follows:

- the operation o handles reassembled products like final function testing or packaging;
- other cases,

where

t^{ps}	possible start time of an operation o (def. VIII.4.17);
P^R	set of remanufactured products (def. VI.2.22);

$c^p_{rao_p}$ (c^p_o)	operation processing cost per unit time (def. VIII.4.8);
$rao_p \in \cup O_p$	reassembly operation of a used product in which a PDSP p belongs;
$t^{ep}_{rao_p, nd_{rap_p}, nq_{rap_p}}$ ($t^{ep}_{o,iq,oq}$)	estimated processing time of rao_p for the non defective input/output quality case (def. VIII.4.20);
$rap_p \in P^R$	remanufactured product in which p is a component (def. VIII.4.21);
$o^s_{rap_p}$ (o^s_p)	start operation of rap_p (def. VI.2.26);
nd_{rap_p} (nd_p)	no defect quality of rap_p (def. VIII.4.22);
$t^{era} \in \mathbb{T}$	estimated reassembly completion time, (Def. VIII.4.25)

where

iq	input quality of p before o processing;
O_p	set of operations (def. VI.2.25).

The reassembled product processing and disposal cost estimation function $f^{RM,edpc\text{-}r}$ and the non reassembled PDSP processing and disposal cost estimation function $f^{NRM,edpc\text{-}r}$ will be defined later.

The first condition is for the case of the post-processing of the reassembled used product which is handled by $f^{RM,edpc\text{-}r}$; it has no complicated calculation but just returns the accumulated cost for the all following post-processing operations. The second condition is for the processing of the used product or subassembly before reassembly. It requires more complicated calculation, because it should be disassembled into composing PDSPs and reassembled again. The operation processing and PDSP disposal/substitution cost of such a case should be estimated for each disassembled PDSPs and summed because of parallel processing of the operations in-between disassembly and reassembly, while the cost for the process after reassembly including the reassembly operation should be estimated only once based on the estimated reassembly ready time of the disassembled PDSPs from p. $f^{NRM,edpc\text{-}r}$ handles the cost before reassembly and $f^{RM,edpc\text{-}r}$ handles the cost after reassembly. Although the part is no more disassembled, it can also be calculated following the second condition by neglecting the disassembly case.

The two functions $f^{RM,edpc\text{-}r}$ and $f^{NRM,edpc\text{-}r}$ have similar forms as follows:

$$f^{RM,epdc-r}(p,o,iq,t^{ps}) = c^w \cdot t^{ew}_o + \sum_{(oq,rto_{oq,n^{rw}},n^{rw}) \in OQ_{o,iq}} \left(rto_{oq} \times \left(c^p_o \cdot t^{ep}_{o,iq,oq} \cdot n^{rw} + f^{RM,cbc}(p,o,iq,oq) \right) \right); \quad \text{(Def. VIII.4.26)}$$

$$f^{NRM,epdc-r}(p,o,iq,t^{ps}) = c^w \cdot t^{ew}_o + \sum_{(oq,rto_{oq,n^{rw}},n^{rw}) \in OQ_{o,iq}} \left(rto_{oq} \times \left(c^p_o \cdot t^{ep}_{o,iq,oq} \cdot n^{rw} + f^{NRM,cbc}(p,o,iq,oq) \right) \right), \text{(Def. VIII.4.27)}$$

where

t^{ps}	possible start time of an operation o (def. VIII.4.17);
c^w	waiting cost per unit time (def. VIII.2.7);

t^{ew}_{o} estimated waiting time (def. VIII.4.5);

$OQ_{o,iq}$ set of operation processing statistics: possible output qualities oq, their frequency ratio $rto_{oq,n^{rw}}$, and the number of operation rework times n^{rw} including the first trial (def. VIII.4.18);

c^{p}_{o} processing cost per unit time (def. VIII.4.8);

$t^{ep}_{o,iq,oq}$ estimated processing time of o for input quality iq and oq (def. VIII.4.20).

The function $f^{NRM,edpt\text{-}r}$ is defined by substituting $f^{RM,cbc}$ with $f^{NRM,cbc}$, which will be explained later in detail. The waiting time cost calculation term $c^{w} \cdot t^{ew}_{o}$ in the function needs no classified calculation, because the output quality does not affect the waiting time for the operation o. The remaining summation term returns the average cost of all output quality cases weighted with the occurrence ratio. The estimated cost for the operation processing and the following process are calculated for each output quality case and weighted averaged together. The estimated cost for the following process is calculated differently depending on cases by the function $f^{RM,cbc}$ and $f^{NRM,cbc}$.

The case by case function for the remanufactured product case $f^{RM,cbc}$ is defined as follows:

$$f^{RM,cbc}(p,o,iq,oq) = \begin{pmatrix} 0 & ,if\ o = o^{e}_{p} \\ \underset{no \in \lambda^{fno}_{p,oq}(o)}{avg} f^{RM,epdc-r}(p,no,oq,t^{ec}_{o,iq,oq}) & ,otherwise \end{pmatrix}, \qquad \text{(Def. VIII.4.28)}$$

of which each of the case by case function is for follows:

– the operation o is the end operation of the whole remanufacturing process;
– other cases,

where

o^{e}_{p} end operation of a PDSP p (def. VI.2.27);[1]

$\lambda^{fno}_{p,oq}$ function specifying p's filtered next operations by its oq (def. VIII.4.24);

$f^{RM,edpc\text{-}r}$ remanufactured product processing and disposal cost estimation function (def. VIII.4.26);

$t^{ec}_{o,iq,oq}$ estimated completion time of o for input quality iq and oq (def. VIII.4.19).

The estimated cost for the following process of the remanufactured product's end operation is zero, hence the first condition returns zero. The following process cost of the remanufactured product's general operation is also simply calculated; the second condition returns the average value of the estimated remanufacturing cost over all alternative feasible following processes which is calculated by recalling the remanufactured product processing and disposal cost estimation function $f^{RM,edpt\text{-}r}$.

The case by case function for the non reassembled PDSP case $f^{NRM,cbc}$ is defined as follows:

[1] It corresponds to the end operation of its whole remanufacturing process, because p is the reassembled product

$$f^{NRM,cbc}(p,o,iq,oq) = \begin{cases} c^{d}_{p} & ,if\ oq \in QS^{D}_{p} \\ f^{da,epdc-r}(p,o,oq,t^{ec}_{o,iq,oq}) & ,else\ if\ \tau^{o}_{p}(o) = disassembly \\ -c^{p}_{o} \cdot t^{ep}_{o,iq,oq} & ,else\ if\ o = o^{e}_{p} \\ \underset{no \in \lambda^{fno}_{p,oq}(o)}{avg} f^{NRM,epdc-r}(p,no,oq,t^{ec}_{o,iq,oq}) & ,otherwise \end{cases}, \quad \text{(Def. VIII.4.29)}$$

of which each of the case by case function is for follows:

- the output quality of the PDSP p after the operation o is the quality to be disposed of;
- o is a disassembly operation;
- o is a reassembly operation;
- other cases,

where

c^{d}_{p} disposal cost of a PDSP p (def. VIII.2.11);
QS^{D}_{p} set of p's qualities to be disposed of (def. VIII.4.23);
$t^{ec}_{o,iq,oq}$ estimated completion time of o for input quality iq and output quality oq (def. VIII.4.19);
τ^{o}_{p} operation type specification function (def. VI.2.35);
c^{p}_{o} processing cost per unit time (def. VIII.4.8);
$t^{ep}_{o,iq,oq}$ estimated processing time of o for iq and oq (def. VIII.4.20);
o^{e}_{p} end operation of p (def. VI.2.27);
$\lambda^{fno}_{p,oq}$ function specifying p's filtered next operations by its oq (def. VIII.4.24);
$f^{NRM,edpc-r}$ non reassembled PDSP processing and disposal cost estimation function (def. VIII.4.27).

The PDSPs to be disposed are directly disposed and substituted with new ones (refer to assumption I.2), and no further disassembly or refurbishment is required before the reassembly operation. Hence the first condition just returns the PDSP disposal cost c^{d}_{p}. The cost for the following process of the disassembly operation is calculated by the disassembled PDSPs parallel processing cost calculation function $f^{da,edpc-r}$ which will be defined later; the second condition just calls $f^{da,edpc-r}$. The end operation of the used product/subassembly/part can be either disassembly or reassembly operation, and the disassembly operation is filtered by the second condition. Hence the third condition corresponds to the reassembly operation case. The processing and disposal cost estimation function f^{epdc-r} already added the reassembly operation processing cost.[1] Therefore the reassembly operation processing cost which is added in its parent formula $f^{NRM,edpc-r}$ should be counterbalanced, because leaving the added reassembly operation cost for each PDSP causes the duplicated calculation of the reassembly cost. The last condition is for the general case of any intermediate operations which are not disassembly or reassembly; it returns the average over all alternative feasible following processes.

The disassembled PDSPs parallel processing cost calculation function $f^{da,edpc-r}$ which is used for the disassembly

[1] The second condition of the processing and disposal cost estimation function f^{epdc-r} includes the reassembly cost (refer to def. VIII.4.25).

operation case of $f^{NRM,cbc}$ is defined as follows:

$$f^{da,edpc-r}(p,o,o\ qt^{ec}{}_{o,iq,oq}) = \sum_{dp \in \lambda^{d}(o)} \begin{cases} c^{d}{}_{dp} & ,if\ \lambda^{cq}{}_{p,dp}(oq) \in QS^{D}{}_{dp} \\ f^{NRM,edpc-r}(dp, o^{s}{}_{dp}, \lambda^{cq}(oq,dp), t^{ec}{}_{o,iq,oq}) & ,otherwise \end{cases}, \text{(Def. VIII.4.30)}$$

of which each of the case by case function is for follows:

- the output quality of a disassembled PDSP p after the operation o is the quality to be disposed of;
- other cases,

where

λ^{d}	disassembly/reassembly relationship function (def. VI.2.23);
$c^{d}{}_{dp}$ ($c^{d}{}_{p}$)	disposal cost of a disassembled PDSP dp (def. VIII.2.11);
λ^{cq}	quality mapping function among PDSPs (def. VI.2.24);
$QS^{D}{}_{dp}$ ($QS^{D}{}_{p}$)	set of dp's qualities to be disposed of (def. VIII.4.23);
$o^{s}{}_{dp}$ ($o^{s}{}_{p}$)	start operation of dp (def. VI.2.26);
$f^{NRM,edpc-r}$	non reassembled PDSP processing and disposal cost estimation function (def. VIII.4.27).

$f^{da,edpc-r}$ returns the summation of all disassembled PDSPs following processing cost, where each disassembled PDSP following processing cost is calculated differently depending on its quality. The first condition is for the disassembled PDSP having the quality to-be-disposed-of. The PDSP to-be-disposed-of is not handled any more and substituted with new one which is directly sent to the buffer for reassembly, hence the fist condition returns the disposal/substitution cost of the PDSP. The following processing cost for the other disassembled PDSP is calculated by recalling its parent function: non reassembled PDSP processing and disposal cost estimation function $f^{NRM,edpt-r}$.

The estimated reassembly completion time t^{era} (def. VIII.4.25) is required for the calculation of the processing and disposal/substitution cost estimation function f^{epdc-r}. t^{era} is calculated by the following reassembly completion time estimation function f^{era-r}:

$$f^{era-r}(p,o,iq,t^{ps}) = \sum_{\substack{(oq,rto_{oq},n^{rw},n^{nw}) \\ \in OQ_{o,iq}}} rto_{oq} \times \begin{cases} f^{era-r}(p, o^{e}{}_{p}, nd_{q}, t^{ec}{}_{o,iq,oq}) & ,if\ oq \in QS^{D}{}_{p} \\ f^{da,era-r}(p,o,oq,t^{ec}{}_{o,iq,oq}) & ,else\ if\ \tau^{o}{}_{p}(o) \\ & \quad = disassembly \\ t^{ec}{}_{o,iq,oq} & ,else\ if\ o = o^{e}{}_{p} \\ \underset{no \in \lambda^{fno}{}_{p,oq}(o)}{avg} f^{era-r}(p,no,oq,t^{ec}{}_{o,iq,oq}) & ,otherwise \end{cases}, \quad \text{(Def. VIII.4.31)}$$

of which each of the case by case function is for follows:

- the output quality of a PDSP p after the operation o is the quality to be disposed of;
- o is a disassembly operation;

– o is a reassembly operation;
– other cases,

where

$OQ_{o,iq}$ set of operation processing statistics: possible output qualities oq, their frequency ratio $rto_{oq,n^{rw}}$, and the number of times of operation processing n^{rw} including the first trial (def. VIII.4.18);
o^e_p end operation of a PDSP p (def. VI.2.27);
nd_q no defect quality of p (def. VIII.4.22);
$t^{ec}_{o,iq,oq}$ estimated completion time of o for input quality iq and output quality oq (def. VIII.4.19);
QS^D_p set of p's qualities to be disposed of (def. VIII.4.23);
τ^o_p operation type specification function (def. VI.2.35);
$\lambda^{fno}_{p,oq}$ function specifying p's filtered next operations by its oq (def. VIII.4.24).

The disassembled PDSPs' estimated reassembly completion time calculation function $f^{da,era\text{-}r}$ will be defined later. $f^{era\text{-}r}$ returns the average value of the estimated reassembly completion time over all output quality cases weighted by the output quality occurrence ratio, and the estimated reassembly completion time for each output quality case should be calculated differently by four cases. The first condition is for the case of the output quality to be disposed of; the disposed PDSP is directly substituted with the new one and sent to the buffer for the reassembly operation, hence the estimated reassembly completion time is calculated by calling $f^{era\text{-}r}$ with the estimated operation completion time $t^{ec}_{o,iq,oq}$ and no defect PDSP quality nd_q as the possible reassembly operation start time and input PDSP quality respectively. The second condition is for the disassembly operation case and returns the maximum value among the estimated reassembly completion time of disassembled PDSPs by calling the disassembled PDSPs' estimated reassembly completion time calculation function $f^{da,era\text{-}r}$. The estimated reassembly completion time is the same with $t^{ec}_{o,iq,oq}$ in case the operation is a non-disassembly end operation of the PDSP p.[1] Hence the third condition just returns $t^{ec}_{o,iq,oq}$. The last condition is for the general case; the estimated reassembly completion time is calculated by averaging the estimated reassembly completion time for alternative next operation selection cases.

The disassembled PDSPs' estimated reassembly completion time calculation function $f^{da,era\text{-}r}$ is defined as follows:

$$f^{da,era-r}(p,o,oq,t^{ec}_{o,iq,oq}) = \max_{dp\in\lambda^d(o)} \begin{cases} t^{ec}_{o,iq,oq} & ,if\ \lambda^{cq}(oq,dp) \in QS^D_{dp} \\ f^{era-r}(dp, o^s_{dp}, \lambda^{cq}(oq,dp), t^{ec}_{o,iq,oq}) & ,otherwise \end{cases}, \quad \text{(Def. VIII.4.32)}$$

of which each of the case by case function is for follows:

– the output quality of a PDSP p after the operation o is the quality to be disposed of;
– other cases,

[1] The end operation of used products or subassemblies is the disassembly operation as discussed above in this section, hence the remaining case after filtering by the second case of the end operation is definitely the reassembly operation.

where

p	PDSP before disassembly operation;
dp	one of p's component after disassembly;
λ^{d}	disassembly/reassembly relationship function (def. VI.2.23);
λ^{cq}	quality mapping function among PDSPs (def. VI.2.24);
QS^{D}_{dp} (QS^{D}_{p})	set of a disassembled PDSP dp's qualities to be disposed of (def. VIII.4.23);
$f^{era\text{-}r}$	reassembly completion time estimation function (def. VIII.4.31);
o^{s}_{dp} (o^{s}_{p})	start operation of dp (def. VI.2.26).

The estimated reassembly completion time of the PDSP to be disposed of is the estimated operation completion time in similar manner to the disassembled PDSPs parallel processing cost calculation function $f^{da,edpc\text{-}r}$ (def. VIII.4.30). The reassembly completion time in the other case is estimated by recalling its parent function $f^{era\text{-}r}$. $f^{da,era\text{-}r}$ is not explained in detail because of its similar definition manner with $f^{da,edpc\text{-}r}$.

4.3. Resource selection

The PDSP selects a resource simultaneously during its operation selection by comparing the estimated performance of operation/resource selection combination cases. Hence, the processing and disposal cost estimation function $f^{epdc\text{-}r}$ and estimated remanufacturing completion time calculation function $f^{ect\text{-}r}$ (refer to figure VIII.5) are extended as follows:

$$f^{Eepdc-r} = \min_{r \in \lambda^{ra^{-1}}(\lambda^{wa}(o))} f^{epdc-r}{}_{r}; \qquad \text{(Def. VIII.4.33)}$$

$$f^{Eect-r} = \min_{r \in \lambda^{ra^{-1}}(\lambda^{wa}(o))} f^{ect-r}{}_{r}, \qquad \text{(Def. VIII.4.34)}$$

where

λ^{ra}	belonging workstation specification function (def. VI.2.11);	
λ^{wa}	workstation in charge specification function (def. VI.2.41);	
$f^{edpc\text{-}r}{}_{r}$	recursive function $f^{edpc\text{-}r}$ (def. VIII.4.14) of the resource r selection case;	(Def. VIII.4.35)
$f^{ect\text{-}r}{}_{r}$	recursive function $f^{ect\text{-}r}$ (def. VIII.4.15) of the resource r selection case.	(Def. VIII.4.36)

The resource to request the operation processing is also selected during the operation selection by applying the extended recursive functions.

4.4. Operation and resource reselection

After a PDSP agent selects an operation to do and a resource to process the selected operation, it registers the selected operation to the selected resource agent's waiting queue and waits until the resource agent announces

the PDSP's processing turn (refer to steps 6 and 16 respectively in figure VIII.4). But in case the waiting can be longer than expected to an undesirable level, the PDSP agent should reselect the operation and resource, because there can be operation and resource combinations having better expected performance.

This research sets a waiting time tolerance to cope with the problem. The PDSP cancels the operation processing request when the operation starting is delayed over the tolerance, and reselects the operation and resource. The tolerance T^{wt}_{p} for the PDSP p which requested an operation o to a resource is defined as follows:

$$T^{wt}_{p} = K^{wt} \times v^{ewt}_{o}, \qquad \text{(Def. VIII.4.37)}$$

where

$K^{wt} \in \mathbb{R}^{+}$	constant coefficient for waiting tolerance.	(Def. VIII.4.38)
v^{ewt}_{o}	variance of estimated waiting time (def. VIII.4.6)	

The EWT is a statistical value derived from the historical realized waiting time. It means the usual waiting time has not a definite value but in a certain range. A general approach of setting the range of a statistical value is utilization of its variance, and this thesis selects the simplest method; this thesis multiplies a constant K^{wt} to the EWT variance v^{ewt}_{o} and sets the range of EWT as $\pm K^{wt} \times v^{ewt}_{o}$ of the average EWT t^{ew}_{o} (def. VIII.4.5). The K^{wt} should be set as the most effective value for system performance. The K^{wt} can be different from QRS to QRS, but this thesis set 2.58 as K^{wt} which is usually used to enclose 99% occurrence cases in normal distributions.

5. Batch processing handling

Many cleaning and refurbishing operations in the remanufacturing system are processed in a batch manner. Hence the scheduling mechanism should embody a PDSP dispatching mechanism for batch resources. The batch processing is a trade-off problem. The batch processing with too small PDSPs wastes the unfilled batch resource capacity and consequently increases the operation cost. In addition, the PDSPs arrived after the operation start loose the chances to shorten their waiting time. On the contrary, the batch processing with too many PDSPs causes early arrived PDSPs' long waiting time in the waiting queue. Therefore a batch processing mechanism should be developed to prevent both the resource operation cost waste and the PDSP waiting time waste.

The batch processing scheduling problem at the certain time is usually divided into two sub problems as follows (Van der Zee, 2004):

– should the batch processing start now or wait more?
– which PDSPs should be dispatched in case of batch start?

This thesis solves each problem as follows:

– start a batch processing when a waiting queue state variable exceeds a preset threshold;
– select the highest priority PDSPs group and then dispatch the higher priority PDSPs in the selected group based on the dispatching rule applied to the resource.

The solution for the first problem is motivated from the minimum batch size rule (Neuts, 1967). While the minimum batch size rule is analytically calculated, the threshold in this thesis is selected by a simulation approach. The consecutive subsections discuss each solution in detail.

5.1. Decision of batch process starting time

The two variables for the batch processing start are defined as follows:

$$s^{wq}_{r} = wt^{acf} \times avg_{O^B \in OCS_r} f^{cfr}\left(r, O^B, WP_r\right) + (1 - wt^{acf}) \times max_{O^B \in OCS_r} f^{cfr}\left(r, O^B, WP_r\right)$$

waiting queue state variable of the batch resource r; (Def. VIII.5.1)

$t^{sb}_{r} \in \mathbb{R}^{+} \leq 1$ threshold for batch process starting of r, (Def. VIII.5.2)

where

$wt^{acf} \in \mathbb{R}^{+} \leq 1$ weight for the average capacity fill up rate of possible simultaneous process-able operation combinations; (Def. VIII.5.3)

$OCS_r = \{O^B \mid \exists(r, O^B, f^{bc}_{r,O^B}) \in v^{bc}\}$ set of simultaneously process-able operation combinations O^B by the resource r; (Def. VIII.5.4)

$WP_r \subset S^{PDSP}$ set of PDSPs in the r's waiting queue; (Def. VIII.5.5)

$f^{cfr}: R \times \cup OCS_r \times S^{PDSP} \rightarrow \mathbb{R}^{+}$ capacity fill up rate calculation function, (Def. VIII.5.6)

where

O^B simultaneously process-able operation combinations set (def. VI.2.44);

f^{bc}_{r,O^B} batch capacity calculation function (def. VI.2.45);

v^{bc} set of batch capacity calculation function specifications (def. VI.2.43);

S^{PDSP} set of PDSPs which are currently in the QRS; (Def. VIII.5.7)

R set of resources (def. VI.2.9).

The waiting queue state variable s^{wq}_{r} is composed of two parts: average capacity fill up rate and each operation set capacity fill up rate. The second term is for the capacity fill up rate of each simultaneously process-able operation combination O^B, and the first term is for the average of them. The various combinations of operations can be simultaneously processed by a batch resource, and the proposed formula is to reflect the variety of combinations as well as the whole. This thesis leave the capacity fill up rate calculation function f^{cfr} as a free

form for the same reason with the batch capacity calculation function f^{bc}_{r,O^B}.

As discussed above, the batch process starts when the waiting queue state variable s^{wq}_r exceeds the threshold t^{sb}_r. The eligible average capacity fill up rate weight wt^{acf} and threshold t^{sb}_r are selected by the whole remanufacturing system simulations for each wt^{acf} and t^{sb}_r combination; this thesis runs pilot simulations for (wt^{acf}, t^{sb}_r) cases of (0.0, 0.0), (0.0, 0.1), (0.0, 0.2), …, (1.0, 0.9), (1.0, 1.0), and selects the values of the case showing the best performance.

5.2. PDSP selection for batch processing

The batch resource to start batch processing selects PDSPs to be processed by the following two steps:

– select the highest priority operation group among the elements of the simultaneously process-able operation combinations set OCS_r (def. VIII.5.4) by the batch resource r;
– select higher priority PDSPs among the PDSPs which requested operations in the selected operation group.

The priority $pri_{oB,r}$ of each simultaneously process-able operation combination O^B (def. VI.2.44) in OCS_r is the average priority of the PDSPs which requested operations in O^B, and it is calculated by the following formula:

$$pri_{O^B,r} = \sum_{p \in WP_{r,O^B}} pri_{p,r} \Big/ |WP_{r,O^B}|, \qquad \text{(Def. VIII. 5.8)}$$

where

$pri_{p,r} \in I^+$ priority of a PDSP p in the batch resource r; (Def. VIII.5.9)

$WP_{r,O^B} = \{p \mid f^{ro}_r(p) \in O^B\} \subset WP_r$ set of PDSPs which requested operations in O^B to r, (Def. VIII.5.10)

where

$f^{ro}_r: WP_r \rightarrow \cup O_p$ function specifying requested operation by PDSP p; (Def. VIII.5.11)

WP_r set of PDSPs in the r's waiting queue (def. VIII.5.5),

where

O_p set of operation of p (def. VI.2.25).

The batch resource calculates each simultaneously process-able operation combination's priority $pri_{O^B,r}$ and selects the highest priority operation group. The definition of the PDSP priority $pri_{p,r}$ represents that the PDSP priority is a positive integer value which indicates the order among the PDSPs in the waiting queue; for instance,

in case three PDSPs p_1, p_2, and p_3 in the waiting queue of a batch resource r_1 respectively have 0.3, 0.2, and 0.6 as their calculated values by the formula of the allocated dispatching rule which gives higher priority to the PDSP having lower calculated value, the priorities of three PDSPs pri_{p_1,r_1}, pri_{p_2,r_1}, and pri_{p_3,r_1} are respectively 2, 1, and 3.

The PDSP selection among the PDSPs which requested operations in the selected operation group is simple; select as many PDSPs as possible in the sequence of PDSP priorities under the batch resource capacity. Hence the set of PDSPs to be dispatched by the batch resource r is defined as follows:

$$PD_r = \{p \mid p \in WP_{r,O^{B-S}}\}, \quad \text{(Def. VIII.5.12)}$$

where

$WP_{r,O^{B-S}}$ set of PDSPs which requested operations in O^{B-S} to r (def. VIII.5.10);

$O^{B-S} \in OCS_r$ selected operation group,

and it should satisfy the following constraints:

– $f^{cfr}(r, O^{B-S}, PD_r) \leq 1$; (Constraint VIII.5.1)

– $\sim\exists p_{ns} \in (WP_{r,O^{B-S}} - PD_r)$: $pri_{p_{ns}} < max_{p \in PD_r} pri_p$; (Constraint VIII.5.2)

– $\sim\exists p_{ns} \in (WP_{r,O^{B-S}} - PD_r)$: $f^{cfr}(r, O^{B-S}, PD_r \cup p_{ns}) \leq 1$, (Constraint VIII.5.3)

where

f^{cfr} capacity fill up rate calculation function (def. VIII.5.6),

Each above constraint sequentially means the selected PDSPs should not exceed the batch resource capacity, no unselected PDSP has the higher priority than any selected PDSPs, and adding any unselected PDSP to the selected PDSP set PD_r makes PD_r exceeds the capacity.

6. Dynamic dispatching rule allocation

6.1. Dispatching rule selection mechanism

The dispatching rule of each resource is dynamically selected and allocated depending on system states by two agents: the system variable monitor agent and the dispatching rule allocator agent (refer to section V.2.1.4 and figure V.4). This research allocates the same dispatching rule to the resources in the same workstation based on the expectation that the best dispatching rule is probably similar for all the resources in an identical workstation.

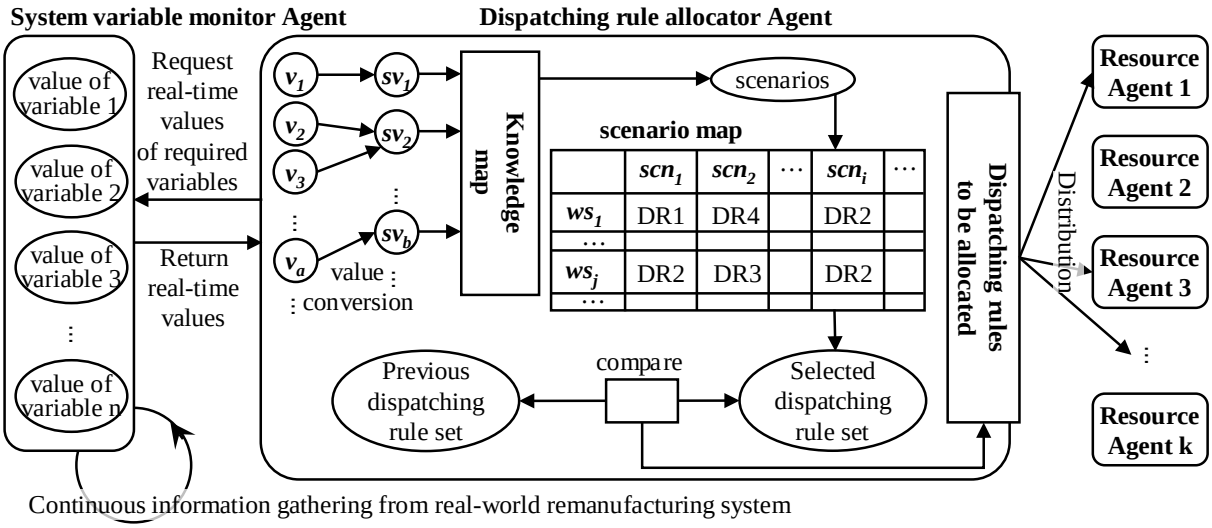


Figure VIII.6. Dispatching rule selection mechanism.

This thesis applies a knowledge-based approach for the dispatching rule selection. The knowledge-based approach can be utilized to solve the problem which is computationally infeasible or difficult to define with mathematical formulations. Hence a knowledge-based approach can be useful for real-time remanufacturing/ manufacturing scheduling because of the following reasons; the remanufacturing/manufacturing system's stochastic characteristics make it realistically impossible to derive a mathematical formula, and even the simulation approach requires too much computation time to get solutions in real-time. This thesis accumulates knowledge by simulations, but the simulations for knowledge accumulation also requires much computation time. Hence a heuristic method to reduce the knowledge accumulation time is proposed in section VIII.6.3.

The knowledge-based remanufacturing/manufacturing scheduling can be executed only when the proper knowledge for the system specification is accumulated and mounted on the QRS, and the updated knowledge is required in case of QRS change; for example, new used product type is additionally remanufactured, facilities are added or removed, and so on. Hence proxy knowledge is required to cover the knowledge accumulation time period for the changed system specifications. This thesis sets the proxy knowledge as the existing knowledge with allocation of the most frequently allocated dispatching rule to new added facilities.

The updated knowledge is mounted to the virtual system without suspending the remanufacturing system; the simulator agent sends messages on the accumulated knowledge to the dispatching rule allocator agent (refer to section V.2.1.4). The resource agents passively change their allocated dispatching rules only when the dispatching rule allocator agent sends dispatching rule change messages. Hence the resource agents can continuously dispatch PDSPs without disturbance from knowledge mounting. This feature is one of the strong points of the multi-agent system.

Figure VIII.6 represents the overall structure of the best dispatching rule allocation process which is as follows:

– request required system values to the system variable monitor agent;
– calculate the values of system state variables which are the input of the knowledge map;
– get a scenario from the knowledge map;

– get the best dispatching rule set for the workstations from the scenario map;
– filter out the workstations requiring no dispatching rule change by comparing the new selected dispatching rule set with the previous one;
– announce the new selected dispatching rules only to the workstations requiring dispatching rule changes.

The symbols in figure VIII.6 are defined as follows:

$v_a \in \mathbb{R}$	system variable which the system variable monitor agent monitors;	(Def. VIII.6.1)
$sv_b \in \mathbb{R}$	system state variable which is the input of the knowledge map;	(Def. VIII.6.2)
scn_i	scenario of the best dispatching rules set for workstations;	(Def. VIII.6.3)
$ws_j \in W$	workstation on the remanufacturing shop,	

where

W	set of workstation (def. VI.2.8).

The system variable v_a and system state variable sv_b will be explained in the next section VIII.6.2.

The system variable monitor agent continuously collects current system values and keeps them inside of itself to support the dispatching rule allocator agent's request. The transferred values to the dispatching rule allocator agent are converted to applicable variables of the knowledge map; for example, the buffer density balance level can be set by calculating the variance of all buffer densities when the knowledge map requires the value.

The knowledge map in the proposed mechanism is a decision tree. Its leaf nodes contain the dispatching rule set scenario ID, and its arcs contain the divergence conditions composed of system state variables. Figure VIII.7 shows an example of the decision tree where sv_b corresponds to the same symbol in figure VIII.6. Therefore the consecutive comparison of converted values sv_b with the tree divergence condition leads to the best scenario for the current system state. The construction of the decision tree will be discussed in section VIII.6.3.

The detailed information for a selected scenario from the knowledge map is gathered from the scenario map. The scenario in this thesis means the set of dispatching rules to be allocated to each workstation, hence the scenario map MAP^{SCN} is defined as the following two dimensional matrix of workstations and scenarios (refer to the scenario map part in figure VIII.6):

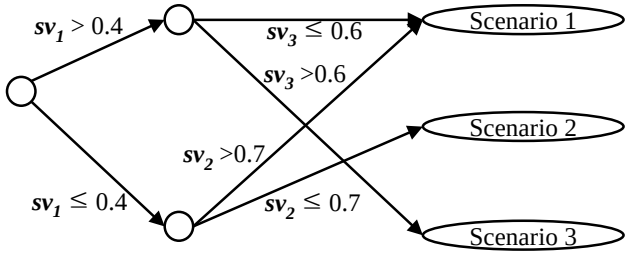


Figure VIII.7. An example of the decision tree in the proposed scheduling mechanism.

$MAP^{SCN} := (dr_{i,j})_{i=1,2,\ldots,s,\ j=1,2,\ldots,w}$; (Def. VIII.6.4)

where

$dr_{i,j} \in DR^A$ dispatching rule to be allocated to the workstation $ws_j \in W$ in case of the ith scenario is selected.
s number of scenarios which are the results of the knowledge map;
w number of workstations in the QRS,

where

DR^A available dispatching rule set (def. VIII.4.3);
W workstation set (def. VI.2.8).

Hence the dispatching rule to-be-allocated to workstation ws_j is the dispatching rule $dr_{i,j}$ when an ith scenario is selected by the knowledge map. The new selected dispatching rules are not announced to the workstations which currently apply the same dispatching rule with the new selected one to reduce the amount of communications.

6.2. System state variables

The system state variables are the input of the knowledge map in the dispatching rule allocator agent (refer to figure VIII.6), and the dispatching rule of each workstation is allocated depending on the system state variables. Hence the system state variables used as the knowledge map input should be able to distinguish the different system states well to show different performance depending on the applied dispatching rules to the workstations.

The reasonable and meaningful system state variables can be different from QRS to QRS. Hence this thesis just suggests some candidate system state variables as follows:

– sv^{pu}: degree of processing urgency; (Def. VIII.6.5)
– sv^{pp}: degree of remanufacturing process progress; (Def. VIII.6.6)
– sv^{ru}: degree of resource utilization, (Def. VIII.6.7)

and each variable is defined as follows:

$$sv^{pu} = \underset{p \in S^{PDSP}}{avg}\left(t^{rp}_{p} \Big/ max(t^{d}_{p} - t^{current}, t^{rp}_{p}) \right);$$

$$sv^{pp} = \underset{p \in S^{PDSP}}{avg}\left(1 - t^{rp}_{p} \Big/ t^{wp}_{p} \right);$$

$$sv^{ru} = \underset{r \in R}{avg}\left(rt_{r} \Big/ ts \right),$$

where

S^{PDSP}	set of current existing PDSPs in the QRS (def. VIII.5.7);	
t^{rp}_p	remaining remanufacturing processing time at current time (def. VII.1.9);	
$t^{current}$	current time (def. VII.1.8);	
t^{d}_p	due date (def. VII.1.7);	
$t^{wp}_p \in \mathbb{T}$	whole remanufacturing processing time of the PDSP p from entering the remanufacturing shop to its remanufacturing completion;	(Def. VIII.6.8)
$ts \in \mathbb{T}$	time slice decided by the person in charge of the QRS control;	(Def. VIII.6.9)
$rt_r \in \mathbb{T}$	running time of resource r during the time slice ts.	(Def. VIII.6.10)

The degree of processing urgency sv^{pu} is the average of each PDSP's remaining process urgency which means the ratio of the remaining operations processing time t^{rp}_p to the remaining time to the due date $t^{d}_p - t^{current}$. This thesis substitutes $t^{d}_p - t^{current}$ with $max(t^{d}_p - t^{current}, t^{rp}_p)$ to prevent the 0 denominator and limit the maximum value of sv^{pu} to 1. The sv^{pu} close to 1 means PDSPs mostly have no spare time to complete their remaining processes in their due dates, and the value close to 0 means the contrary case: most of PDSPs have enough spare time.

The degree of remanufacturing process progress sv^{pp} is the average of each PDSP's remanufacturing completion rate which means the ratio of the processed operations' processing time to the whole remanufacturing processing time. Each PDSP's remanufacturing completion rate is inversely calculated by subtracting the remaining incompletion ratio from 1, because the processed operation's processing time cannot be coordinated with the whole remanufacturing processing time; this is the estimated value from operation time statistics but that is the realized value. sv^{pp} close to 1 means PDSPs are mostly at the later part of their remanufacturing processes, and the value close to 0 means most of PDSPs just entered into the remanufacturing system.

The degree of resource utilization sv^{ru} is the average of resources' running time during a certain period. sv^{ru} close to 1 means many PDSPs are in the QRS, and the value close to 0 means that not so many PDSPs are in the QRS.

The system state distinguish-ability of each candidate system state variable is probably different depending on the QRS specifications. Hence the system variables should be selected differently depending on a case by case; they can be selected by simulations or the field experts' insight. The examples of system state variable selection can be found in the scheduling mechanism validation section VIII.7.2. The QRS can also consider other candidate system state variables which are derived by the field experts' experiences.

6.3. Knowledge generation by simulation with a heuristic approach

The knowledge map is a decision tree which indicates the scenarios on the best dispatching rule set of workstations depending on the system states represented by the system state variables. Hence the decision tree in the knowledge map can be constructed from the raw information on the best dispatching rule set of workstations for each system state case which is gathered by simulations. Although the simplest way to find the set is to simulate all possible combinations of dispatching rules for workstations, such an approach has a complexity of $O(n^m)$ where n and m are respectively the numbers of dispatching rules and workstations, and the algorithms of such complexity by the full enumeration are realistically incalculable in a reasonable time for large problem cases. This section discusses a heuristic approach to cope with the problem.

The best dispatching rule set for each system state is found by the following five steps instead of the full enumeration:

a. Find the default dispatching rule;
b. Classify workstations into three groups: disassembly, refurbishment, and reassembly;
c. Classify dispatching rules into some groups depending on their performance;
d. Find the best dispatching rule group for each of the three workstation groups;
e. Find the best dispatching rule for each workstation group.

The main point of this approach is the classifications of workstations and dispatching rules. The simulation times can be greatly reduced compared to the full enumeration by finding the best dispatching rule group first and the best dispatching rule among the found dispatching rule group later. This thesis also classifies workstations into three groups and allocates the same dispatching rule to the workstations in the same group, which also reduces the simulation times greatly. The consecutive subsections explain each step.

This thesis also proposes another more sophisticated heuristic method without workstation grouping, because the better performance is expected when allocating a different dispatching rule to each workstation even in the same group. Although the heuristic without workstation grouping requires less simulation times than the full enumeration, it is still not short enough. Hence this thesis just leaves the heuristic method without workstation grouping in appendix I for the case of using a very high performance simulation system and does not utilize it any more.

6.3.1. Finding the default dispatching rule (step a)

The default dispatching rule is selected in a simple way as follows:

– simulate and get the performances of each candidate dispatching rules;
– select the dispatching rule that shows the best performance.

Therefore the default dispatching rule can be different depending on the performance measures to be maximized. This thesis select the default dispatching rule with the total remanufacturing cost performance measure $\boldsymbol{M^{perf}}$ (def. VIII.2.1). In case the measure is inapplicable because of lack of information, conventional performance measures like mean flow time or tardiness can be alternatively used.

6.3.2. Workstation classification (step b)

The workstations are classified into three groups which are motivated from the idea that the remanufacturing process sequentially follows the three steps roughly: the used products and subassemblies are disassembled, the disassembled parts are refurbished, and the disassembled parts are reassembled. Hence this thesis classifies the workstations into the following three groups:

– disassembly group: workstations mainly handle used products and subassemblies as input PDSPs;
– refurbishment group: workstations mainly handle disassembled parts as input PDSPs;
– reassembly group: the reassembly workstations and the post-reassembly workstations mainly handle reassembled products as input PDSPs.

Even though the above classification is usually uncomplicated, there can be exceptional workstations which can belong to two groups; there is an actual case of a cleaning workstation which cleans a used product before disassembly as well as a part disassembled from a different used product (Sakara, 2006). This thesis classifies this type of workstation $\boldsymbol{w}$ depending on its belonging tendency $\boldsymbol{bt_{wg,w}}$ to each workstation group $\boldsymbol{wg}$, the calculation function of which is defined as follows:

$$\boldsymbol{bt_{wg,w}} = \sum_{p \in P_{wg,w}} \boldsymbol{avg(ls_{up_p})} \Big/ \boldsymbol{avg(ai_{up_p})}, \qquad \text{(Def. VIII.6.11)}$$

where

$\boldsymbol{P_{wg,w} \subset P}$	set of PDSPs which is handled by the workstation $\boldsymbol{w}$ and belong to the PDSP group of $\boldsymbol{wg}$ classification criteria;	(Def. VIII.6.12)
$\boldsymbol{ai_{up_p}}$ ($\boldsymbol{ai_p}$)	used product arrival interval statistics (def. VI.2.36);	
$\boldsymbol{ls_{up_p}}$ ($\boldsymbol{ls_p}$)	used product arrival lot size (def. VI.2.37);	
$\boldsymbol{up_p \in P^U}$	used product having the PDSP $\boldsymbol{p}$ as its component,	

where

$\boldsymbol{P}$	set of PDSPs (def. VI.2.20);
$\boldsymbol{P^U}$	set of used products (def. VI.2.21).

For example, the above mentioned cleaning workstation can be classified into the disassembly workstation group in case the number of arrival used products per unit time of the former used product is bigger than latter's.

6.3.3. Dispatching rule classification (step c)

Every dispatching rule has its own focus on the performance measure which it wants to maximize. Hence we can consider grouping the candidate dispatching rules depending on their pursuing performance measures like the following four groups: processing time, due date, work in process, and cost (Ramasesh 1990). But the dispatching rule performance comparison in chapter VII explains that the dispatching rule's similarity o the focusing performance measures does not guarantee their performance similarity.[1] Hence this thesis classifies dispatching rules depending on the similarity of their performance gathered from simulation; for example, the busy system case in section VII.2.3.1 can put the LPOSU, MDD, WINQ, and LPOSM rules in the same group,

[1] For example, the mean tardiness in the busy system of EDD and MDD is 1.455 and 1.054 respectively (refer to figure VIII.4), even though the two dispatching rules focus on the due date measure.

because they shows similar performance in the flow time and tardiness measurement.

The dispatching rule groups are defined as follows:

$\boldsymbol{S^{DRG}} = \{\boldsymbol{DRG} \mid \boldsymbol{DRG} \subset \boldsymbol{DR^A}\}$ set of dispatching rule group sets, (Def. VIII.6.13)

where

$\boldsymbol{DRG}$ set of dispatching rule group, (Def. VIII.6.14)
$\boldsymbol{DR^A}$ available dispatching rule set (def. VIII.4.4).

Dispatching rules should be grouped differently depending on their target performance measure. Hence the classification can be different from QRS to QRS and their target performance measure. The proposed heuristics selects a dispatching rule in a selected dispatching rule group. Therefore every ***DRG*** should be non-empty (Constraint VIII.6.1), and each dispatching rule should belong to at least one ***DRG*** (Constraint VIII.6.2). Consequently the following restrictions should be kept:

– $\forall \boldsymbol{DRG} \in \boldsymbol{S^{DRG}}: \boldsymbol{DRG} \neq \emptyset$; (Constraint VIII.6.1)
– $\forall \boldsymbol{dr} \in \boldsymbol{DR^A}: \boldsymbol{dr} \in \exists \boldsymbol{DRG} \in \boldsymbol{S^{DRG}}$. (Constraint VIII.6.2)

6.3.4. Finding the best dispatching rule group (step d)

The performance of each dispatching rule group allocation to each workstation group is examined by simulations to find the best dispatching rule group set, and the default dispatching rule is allocated to other workstation groups. Hence the complexity of simulation times for each system state case is $\boldsymbol{O}(\boldsymbol{n})$ where $\boldsymbol{n}$ is the number of dispatching rule groups. The dispatching rule group showing the best performance for each workstation group is selected for each system state case based on the simulation results.

The dispatching priority $\boldsymbol{pri^G_{DRG,p,r}}$ of a PDSP $\boldsymbol{p}$ by the dispatching rule group ***DRG*** in a resource $\boldsymbol{r}$'s waiting queue is calculated by averaging the priority by each dispatching rule in ***DRG*** as follows:

$$\boldsymbol{pri^G_{DRG,p,r}} = \sum_{\boldsymbol{dr} \in \boldsymbol{DRG}} \boldsymbol{pri_{dr,p,r}} \Big/ \left|\boldsymbol{DRG}\right|, \quad \text{(Def. VIII.6.15)}$$

where

$\boldsymbol{pri_{dr,p,r}} \in \mathbb{I}^+$ dispatching priority of a PDSP $\boldsymbol{p}$ by a dispatching rule ***dr*** in the $\boldsymbol{r}$'s waiting queue; (Def. VIII.6.16)
DRG set of dispatching rule group (def. VIII.6.14).

6.3.5. Finding the best dispatching rule (step e)

The best dispatching rule in the selected dispatching rule group is selected in the same way as the best

dispatching rule group selection; the performance of the allocation of each dispatching rule in a selected dispatching rule group to each workstation group is examined by simulations. The possible dispatching rules for a workstation group is restricted to the dispatching rules in the selected best dispatching rule group for each workstation group. The dispatching rule combination showing the best performance finally set for each system state case.

7. Validation of the proposed scheduling mechanism

7.1. Distinctive features of the proposed scheduling mechanism

This thesis proposed a QRS scheduling mechanism because of no proper mechanisms which actively schedule a remanufacturing system with quality consideration in real-time. The proposed scheduling mechanism is motivated from the market-based agent negotiation and the dynamic dispatching rule allocation over a multi-agent framework. A batch processing scheduling mechanism is also suggested as a solution of a sub-problem of the remanufacturing system scheduling.

The market-based negotiation is popularly used in the multi-agent manufacturing scheduling. While each job for a product with a static manufacturing process is usually defined as an agent in the previous research (Macchiaroli and Riemma 2002 and Kim *et al.* 1996), this thesis defined all the PDSPs as agents and concretized the method of handling disassembled/reassembled PDSPs which do not exist in the conventional manufacturing system. This thesis proposed a performance measure which is used as an objective function when an agent decides the operations to do and the resources to do the operations. A recursive calculation method for the performance estimation with the objective function is also proposed because of the difficulty in the straightforward calculation.

Several research on the knowledge-based dispatching rule allocation through the periodical system state monitoring are found in the conventional manufacturing domain, but they usually allocate an identical dispatching rule to all the resources in the manufacturing shop (Arzi and Iaroslavitz 1999 and Shiue 2009). This thesis allocates different dispatching rules to different resources through continuous system state monitoring (refer to figure VIII.6). Knowledge-based approaches in previous research require many training samples and the samples usually do not cover all possible input variable spaces, even though they allocate an identical dispatching rule to all resource for a system state. The different dispatching rule allocation to resources in this thesis requires more training samples than other approaches. Hence this thesis proposed a heuristic method to reduce the knowledge generation time. The base information gathered by simulations with the heuristic method for the decision tree creation covers all the ranges of system state variables. Consequently the proposed scheduling mechanism can avoid encountering unexpected results from the uncovered system state variable ranges, that was a week point of general knowledge-based approaches.

The proposed batch processing mechanism is motivated from the minimum batch size rule (Neuts, 1967). Although many previous research focused on the batch resource itself, this thesis finds the best solution by simulations of the whole system. The proposed mechanism considers capacity fill up rate by waiting PDSPs

from the two perspectives: the fill up rate by a simultaneously process-able PDSP group and the fill up rate by whole waiting PDSPs. Hence the mechanism start batch processing with consideration of not only the waiting PDSPs' unbalanced concentration to a specific simultaneously process-able PDSP group but also the balanced states.

The main distinctive features of the proposed scheduling mechanism can be summarized as follows:

- communication protocol for the used product remanufacturing including disassembly and reassembly;[1]
- cost-based integrated performance measure for a QRS;
- recursive calculation method for a PDSP's performance estimation;
- continuous monitoring of system states and knowledge-based system state dependent dispatching rules for resources where the dispatching rule is different from resource to resource;
- heuristic method for generating knowledge which covers whole possible system states;
- batch processing scheduling method covering the waiting PDSPs' unbalanced concentration cases.

7.2. Performance enhancement by the proposed scheduling mechanism

The proposed scheduling mechanism is numerically validated by simulations in this section. The simulation condition is exactly the same with that applied for the proposed dispatching rule evaluation: the total work content (TWK) approach is applied for the due date setting, the main simulation is done with 10 runs with 5,000 used products and the pilot simulation is done with 5 runs with 3,000 used products, operations/resources are randomly selected when the PDSP's operation/resource selection mechanism is not applied, and the batch processing starts instantly even when at least one PDSP is in the waiting queue in case the batch processing mechanism is not applied. Section VII.2.2 explained the experiment design and simulation approach in detail.

The effect of the proposed scheduling mechanism is validated with the Student's t-test.[2] The objective of the t-test in this section is to see the performance improvement compared to the existing scheduling methods as with the dispatching rule comparison in chapter VII, hence the two-tailed t-test is done. All simulations are independent from each other, hence independent unequal distribution is also assumed as the above t-tests in this thesis (refer to sections VI.5.2 and VII.2.3).

7.2.1. Effect in traditional remanufacturing systems without quality information

The proposed scheduling mechanism mainly comprises the following two mechanisms:

- PDSP's operation/resource selection;
- Dynamic dispatching rule allocation.

While the quality information is the major factor in the first mechanism, the second one can involve no quality

[1] Some explanations on handling of the agents for the disassembled/reassembled PDSPs are also found in section V.2.1.3.
[2] This thesis utilized the Microsoft® office excel 2003 for all the mathematical and statistical calculations.

information by selecting system state variables and performance measures which consider no PDSP/resource quality. Hence this section evaluates the effect only of the proposed dynamic dispatching rule allocation mechanism. The following subsections evaluate a mobile phone repair system[1] and an automotive part remanufacturing system which were handled for the extended two-level colored Petri-nets (XCPN) (Sakara 2006) validation. The second case includes batch processing resources, hence the proposed batch processing mechanism is also applied for the validation of it.

7.2.1.1. Mobile phone repair system (case study 1)

7.2.1.1.1. System description

The mobile phone repair system discussed in this section handles five types of mobile phones which are repaired by the following six sequential operations: unpacking, sorting, repair/configuration, personalization, repair report recording, and packing (refer to table VIII.2). The personalization and packing step show deterministic processing time. The others show uniform processing time distribution, and the processing time rage is different from product to product as in table VIII.2. The repair quality is tested after the repair/configuration step, and 25% of the repaired products fail to pass the test and sent back to be repaired again.

Each operation is processed by one workstation on the repair shop and each workstation is composed of only one resource except for the workstations for the repair/configuration operation OP3 and repair report recording operation OP5; workstations for those operations has 7 and 4 resources respectively. The arrival interval of each mobile phone type shows a Poisson distribution with the average of 26 minutes. 1.05 is set for the due date adjustment factor (refer to section VII.2.2).

No quality embedded dispatching rules are selected as candidates because of no quality information in the repair system. Therefore the conventional benchmark dispatching rules and a dispatching rule designed for the remanufacturing system (Sakara 2006) are selected as candidates: the earliest due date (EDD), first come first serve (FCFS), modified due date (MDD), new earliest due date (NEDD), remaining processing time (RPT), shortest processing time (SPT), and work in next queue (WINQ) rules (refer to table VII.1). Table VIII.3 shows the pilot simulation results to find the default dispatching rule. NEDD shows the best performance in the flow time as well as tardiness overall, hence this thesis selects NEDD as the default dispatching rule for the repair system.

Table VIII.2. Mobile phone repair process and processing time of each operation.

Operation		Processing time (minute)				
	Distribution	Distribution parameter of each product				
ID Name	type	A	B	C	D	E
OP1 Unpacking	Uniform	2.0 - 3.0	1.0 - 4.0	2.0 - 3.0	1.0 - 2.0	2.0 - 4.0
OP2 Sorting	Uniform	0.5 - 1.0	1.0 - 2.0	1.0 - 1.5	0.5 - 1.5	0.5 - 2.0
OP3 Repair/configuration	Uniform	20.0 - 27.0	10.0 - 30.0	20.0 - 25.0	15.0 - 30.0	15.0 - 25.0
OP4 Personalization	Deterministic	3.0	2.0	4.0	3.0	4.0
OP5 Repair report recording	Uniform	5.0 - 10.0	7.0 - 8.0	3.0 - 12.0	4.0 - 8.0	7.0 - 12.0
OP6 Packing	Deterministic	3.0	4.0	3.0	2.0	3.0

[1] The repair system is frequently considered as a remanufacturing system because of their similarity; products are disassembled, repaired, and reassembled (refer to the section III.3).

Table VIII.3. Pilot simulation results to find the default dispatching rule for the mobile phone repair system.

	Flow time (minutes)			Tardiness (minutes)			
Dispatching rule	Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[a] (%)
EDD	97.6	455	63.4	53.2	409	62.6	86.6
FCFS	90.0	732	59.5	45.9	688	58.7	84.0
MDD	92.9	810	82.4	48.7	763	81.3	82.9
NEDD	87.7	362	45.5	43.5	318	44.5	84.5
RPT	92.7	910	81.5	48.5	862	80.3	83.2
SPT	95.2	955	85.2	51.0	908	84.2	83.2
WINQ	90.3	1,076	70.7	46.2	1,032	70.1	82.9

[a] PoTJ: Percentage of tardy jobs.

7.2.1.1.2. Knowledge map

The degree of resource utilization sv^{ru} (def. VIII.6.7) among the three introduced state variables (refer to section VIII.6.2) is arbitrarily selected as the input system state variable of the knowledge map. Figures VIII.8 and VIII.9 show the sv^{ru} change and occurrence frequency during 1000 used product repairing by a simulation. sv^{ru} changes actively and moves usually in-between 0.4 and 0.6, hence sv^{ru} is divided into four ranges with the division points of 0.45, 0.5 and 0.55 which roughly match with the accumulated occurrence frequency of 0.25, 0.5, and 0.75 respectively; the divided ranges are as follows:

- sv^{ru} range 1: 0.00 - 0.45;
- sv^{ru} range 2: 0.50 - 0.55;
- sv^{ru} range 3: 0.45 - 0.50;
- sv^{ru} range 4: 0.55 - 1.00.

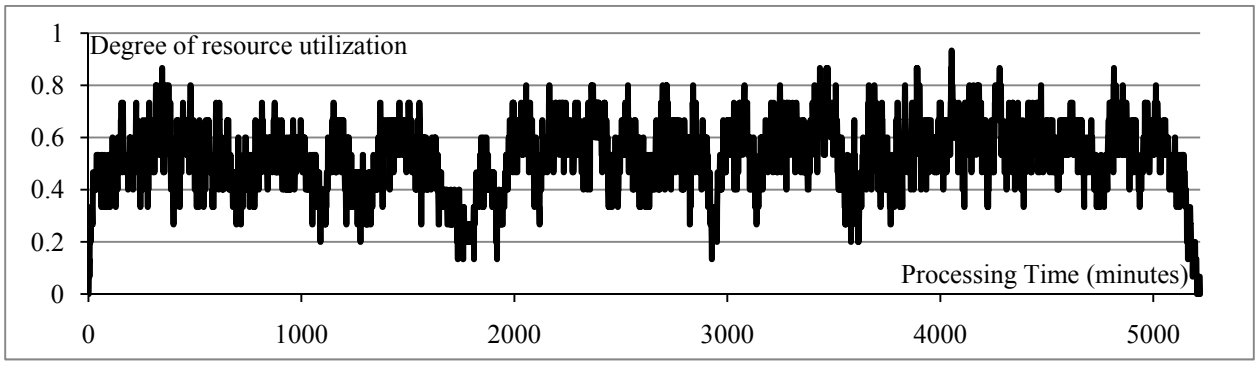


Figure VIII.8. Change of the degree of resource utilization during 1000 mobile phone repairing.

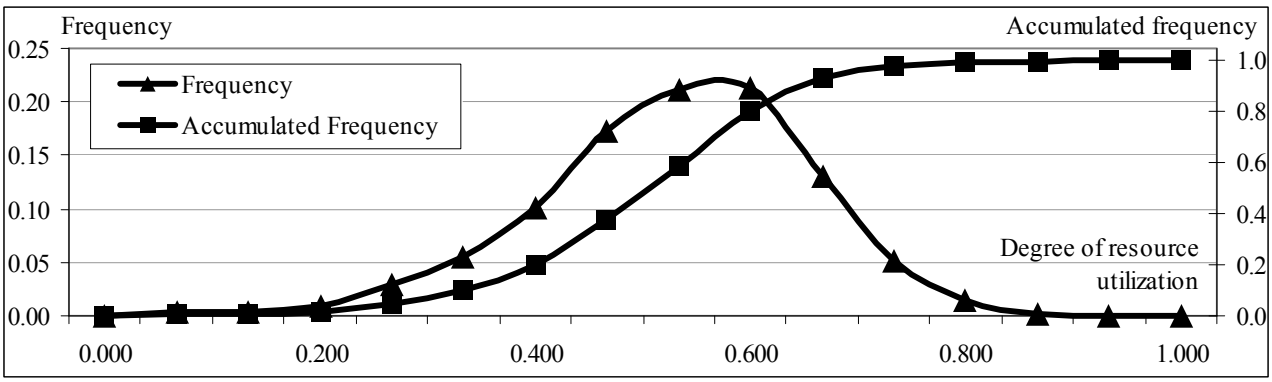


Figure VIII.9. Occurrence frequency of the degree of resource utilization during 1000 mobile phone repairing.

Table VIII.4. Best dispatching rule for each state variable ranges for the mobile phone repair system.

sv^{ru} range	Best dispatching rule	Flow time (minutes)			Tardiness (minutes)			
		Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[a] (%)
Range 1	RPT	86.2	359	48.2	42.0	312	47.1	83.8
Range 3	EDD	84.1	304	41.0	39.9	258	40.0	84.8
Range 4	FCFS	86.9	360	47.1	42.6	316	46.2	84.5
Default dispatching rule: NEDD		87.7	362	45.5	43.5	318	44.5	84.5

[a] PoTJ: Percentage of tardy jobs.

Table VIII.5. Performance of the mobile phone repair system with/without the scheduling mechanism.

Model	Flow time (minutes)			Tardiness (minutes)			
	Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[b] (%)
DDA[a] (α)	88.4	480	51.1	44.2	434	49.6	84.7
NEDD (β)	92.1	390	50.8	48.2	345	50.0	86.2
Improvement[c] (%)	4.01	-22.96	-0.65	8.31	-25.86	0.84	1.74
P-value from T-test	0.050	0.020	0.466	0.043	0.021	0.454	0.009

[a] DR: dispatching rule.
[b] PoTJ: Percentage of tardy jobs.
[c] [improvement] = $(\beta-\alpha)/\beta\times100$; the calculation results can be different with the value calculated with that in the table because of the value in the table is rounded from the original value.

Although the used products are disassembled, repaired, and reassembled on the repair shop, they move together in a bucket, and there is no different process among parts from an identical used product. Hence the repair system in this case can be considered as the used product treatment without any disassembly/reassembly. Consequently this thesis does not classify the workstations but applies the same dispatching rule to all the workstations,[1] because no suitable workstation classification criteria are found. Therefore each scenario indicates just one dispatching rule. No workstation classification has the benefit to reduce the simulation time tremendously, hence all candidate dispatching rules are directly examined for each state variable range without finding the best dispatching rule group. Table VIII.4 represents the selected best dispatching rule for each state variable range from the pilot simulations; range 2 has no better dispatching rule than the default dispatching rule.

7.2.1.1.3. Performance improvement analysis

Table VIII.5 summarizes the simulation results of the two cases: the NEDD rule application case and the dynamic dispatching rule allocation case. It shows that the dynamic dispatching rule allocation improves the repair system performance by 4.01% and 8.31% from the perspective of the mean flow time and tardiness respectively. The p-values from the t-test confirm the performance improvement with about 95% confidence level. The dynamic dispatching rule allocation also improves the percentage of tardy jobs about 1.74%. Even though the improvement is not so much, the p-value explains its definite improvement in 99% confidence level.

On the contrary, no improvement is found in the maximum or standard deviation (STD) performance measures. The tardiness STD is improved only by 0.84%, and the flow time STD is rather aggravated by 0.65%. But the p-value from the t-test shows that the STD difference is meaningless. The dynamic dispatching rule allocation also caused a big definite aggravation of the maximum flow time and tardiness by about 25% in 98% confidence level.

[1] The proposed scheduling mechanism classifies workstations into three groups and tries to apply different dispatching rules to the different workstation groups (refer to section VIII.6.3.2).

Table VIII.6. Workstations in the automotive part remanufacturing system.

ID	Name	Description
WS01	Disassembly 1	disassemble WP
WS02	Thermal cleaning	thermal clean AS and big parts of WP
WS03	Big sandblasting	sandblasting big parts of both WP and AS
WS04	Big sandblasting testing	test sandblasted big parts of both WP and AS
WS05	Painting 1	paint big parts of WP
WS06	Small sandblasting 1	sandblasting small parts of WP
WS07	Water cleaning	water clean small parts of WP
WS08	Water cleaning testing	test water cleaned small parts of WP
WS09	Finishing	finish water cleaned small parts of WP
WS10	Small reassembly 1	reassemble small parts of WP
WS11	Big reassembly 1	reassemble WP
WS12	Packaging	pack remanufactured WP
WS13	Disassembly 2	disassemble AS
WS14	Small sandblasting 2	sandblasting small parts of AS
WS15	Small sandblasting testing	test sandblasted small parts of AS
WS16	Small reassembly 2	reassemble small parts of AS
WS17	Big reassembly 2	reassemble AS
WS18	Painting 2	paint remanufactured AS

In conclusion, the dynamic dispatching rule allocation improves the repair system performance with over 95% confidence level from the perspective of major traditional performance measures; the mean flow time and tardiness. More improvement can be still possible because of the unapplied factor in the proposed dynamic dispatching rule allocation mechanism: the workstation classification. Classifying workstations and allocating different dispatching rules for each group are expected to show better performance. This thesis arbitrarily selected the state variable and only one state variable is used for the system states classification. Hence the adoption and sophisticated range division of state variables can improve further the system performance.

7.2.1.2. Automotive part remanufacturing system (case study 2)

7.2.1.2.1. System description

The automotive part remanufacturing system discussed in this section handles two types of used products: the water pumps [WP] and the axle struts [AS] in a remanufacturing job-shop.

The remanufacturing shop is equipped with 18 workstations (refer to table VIII.6). Each workstation contains one resource and the resources in the thermal cleaning workstation [WS02] and big sandblasting workstation [WS03] are batch resources. The capacities of the thermal cleaning batch resource are 40 units for the big parts of [WP] and 48 units for [AS], and that of the big sandblasting batch resource are respectively 20 and 24 units for the big parts of [WP] and [AS]. Each batch resource can handle only one kind PDSP at one time; for example, the thermal cleaning batch resource cannot clean [AS] and the big parts of [WP] at the same time. In-between workstations exit proper buffers, the capacities of which are big enough to contain all PDSPs handled in the remanufacturing shop at the same time.

The remanufacturing processes of the used products comprise the operations in table VIII.7. Each operation's processing time has a deterministic distribution. The collected [WP] are disassembled into several parts which are classified into big parts [WB] and small parts [WS], and parts in each group are handled together in a bucket.

Hence [WP] can be considered to be disassembled into two parts. The quality of [WB] and [WS] are examined after their sandblasting operation OPw03 and cleaning operation OPw07, and 60% and 70% of them pass the quality tests respectively. The [WB] that failed the first quality test are sandblasted again, and the 25% of the re-sandblasted [WB] pass the test. The [WP] that failed again the second quality test are disposed of directly.

[AS] also has a similar remanufacturing process and the disassembled parts are handled in buckets in the same way with [WP]. The detailed operations and quality test pass ratios are found in table VIII.7.

Table VIII.8 summarizes the used product arrival and order receiving statistics. The order of remanufactured used products is independent with the used product arrival. The arrival intervals of [WP] and [AS] are 40.75 and 35 minutes respectively, and only one used product is collected at each time. The used product arrival distributions of both used products follow a Poisson process. The orders of remanufactured [WP] and [AS] are received every 419 and 300 minutes in average, and the lot size of each order are 7.2 and 5.14 units in average respectively. The order arrival interval and the lot size distribution also follow Poisson processes. The due dates of orders are composed of 40% of one day (1,440 minutes) and 60% of one week (10,080 minutes) from the order receiving time. The due date distribution is valid for both used products, hence this case does not applies the TWK method for due date setting.

The candidate dispatching rules are the same with that of the above mobile phone repair system case in section VIII.7.2.1.1 because of the same reason; no quality information is involved. Table VIII.9 shows the pilot simulation results to find the default dispatching rule. Tardiness related measures show much difference, because its due date is independent of the used product arrival and has high variance. Hence this thesis selects NEDD as the default dispatching rule based on the mean flow time measure. All the other simulation analysis in this section is done based on the mean flow time for the same reason.

Table VIII.7. Operations in the remanufacturing processes of the water pumps and axle struts of the automotive part remanufacturing system.

ID	Name	Processing time	Quality test pass ratio
OPw01	WP disassembly	8.79	
OPw02	WB thermal clean	150	
OPw03	WB sandblasting	4.26	
OPw04	WB sandblasting testing	3.14	1st: 60%, 2nd: 75%
OPw05	WB painting	1.6	
OPw06	WI sandblasting	7.61	
OPw07	WI water cleaning	4.33	
OPw08	WI cleaning testing	4.26	1st: 70%
OPw09	WI finishing	4.26	
OPw10	WI semi-reassembly	4.32	
OPw11	WP reassembly	9.11	
OPw12	WP packaging	6	
OPa01	AS thermal cleaning	210	
OPa02	AS disassembly	10.7	
OPa03	AB sandblasting	3.33	1st: 50%, 2nd: 20%
OPa04	AI cleaning	6.71	
OPa05	AI cleaning testing	1.72	1st: 60%
OPa06	AI preassembling	2.4	
OPa07	AS reassembly	13.73	
OPa08	AS painting	8.11	

Table VIII.8. Used products arrival/order distributions in the automotive part remanufacturing system.

	used product arrival		order receiving		
Used product	interval (minutes)	lot size (units)	interval (minutes)	lot size (units)	due date (minutes)
[WP]	40.75	1.0	419	7.20	40%: 1,440
[AS]	35.00	1.0	300	5.14	60%: 10,080

* All distributions follow Poisson processes, and the indicated numbers in the table are averages.

Table VIII.9. Simulation results for the default dispatching rule of the automotive parts remanufacturing system.

	Flow time (minutes)			Tardiness (minutes)			
Dispatching rule	Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[a] (%)
EDD	803.1	4,253	672.4	205.4	3,911	560.4	0.113
FCFS	660.8	1,881	374.8	470.7	6,805	1,163.0	0.181
MDD	810.0	2,990	596.8	1,194.1	9,150	2,166.3	0.276
NEDD	638.0	3,033	427.9	698.2	9,037	1,798.1	0.204
RPT	716.9	1,938	387.6	1,706.7	11,960	2,905.8	0.310
SPT	779.6	2,208	451.9	1,325.1	10,411	2,422.7	0.308
WINQ	680.7	1,894	380.3	455.3	5,528	1,317.6	0.141

[a] PoTJ: Percentage of tardy jobs.

Table VIII.10. Simulation results for the batch processing parameters of the automotive part remanufacturing system.

	Mean flow time (minutes)				
	Weight of the average capacity fill up rate				
Threshold	0.00	0.25	0.50	0.75	1.00
0.0			638.0		
0.1	619.9	638.9	575.6	603.0	669.5
0.2	743.1	643.3	621.3	703.9	642.4
0.3	796.4	706.8	639.6	731.2	749.9

7.2.1.2.2. Batch processing parameters and knowledge map

This automotive part remanufacturing system has batch resources, hence the proposed batch processing scheduling mechanism is also applied. Although different parameter values for different resources are preferable for the better system performance, this thesis applies the same values to all batch resources for the simplicity.

To select the batch processing parameters of the batch process start threshold t^{sb}_r (def. VIII.5.2) and the average capacity fill up rate weight wt^{acf} (def. VIII.5.3), this thesis run pilot simulations for 10 t^{sb}_r cases with setting 0.5 of wt^{acf} first. Figure VIII.10 show the simulation results and it is found that the mean flow time increases rapidly from 0.4 of t^{sb}_r. Hence 0.1, 0.2, and 0.3 threshold cases are simulated more for different wt^{acf} cases: 0, 0.25, 0.75, and 1. Table VIII.10 represents the pilot simulation results, where the 0.1 t^{sb}_r and 0.5 wt^{acf} combination case shows the minimum mean flow time.

The degree of remanufacturing process progress sv^{pp} (def. VIII.6.6) and the degree of resource utilization sv^{ru} (def. VIII.6.7) are selected as the input system state variable for the knowledge map, because the degree of processing urgency sv^{pu} (def. VIII.6.5) calculation includes the due date; it affects mainly tardiness related measures which are not considered in this case. Figures VIII.11 and VIII.12 represents the change of sv^{pp} and sv^{ru} during 100 remanufactured used products. The state variables are divided in the same way with that applied for the mobile phone remanufacturing system case. While sv^{pp} is divided into four ranges, sv^{ru} is divided into three ranges because of its movement in a comparably small range. Two system state variables are divided as follows:

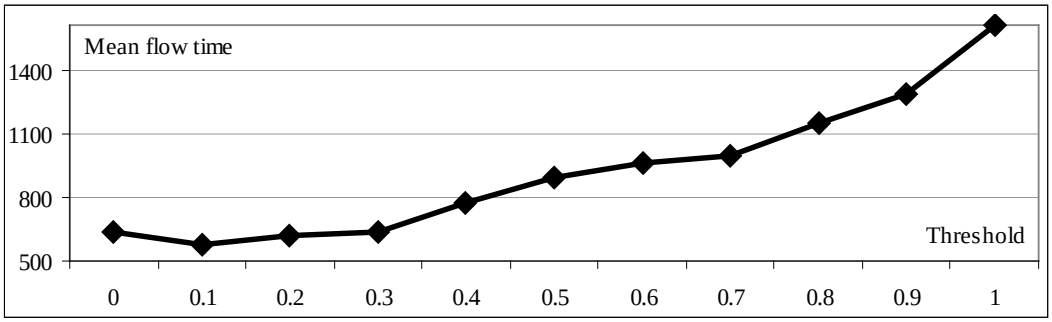


Figure VIII.10. Mean flow time for each batch capacity fill up rate threshold with 0.5 of the average capacity fill up rate weight for the automotive part remanufacturing system.

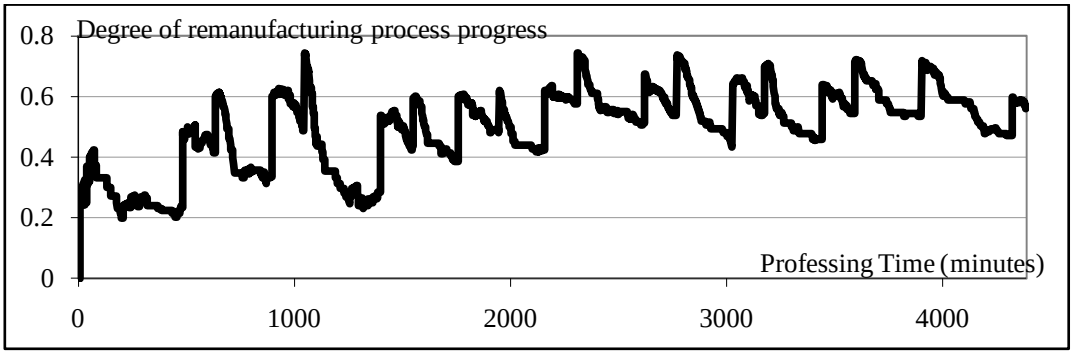


Figure VIII.11. Change of the degree of remanufacturing process progress during 100 remanufactured used products of the automotive part remanufacturing system.

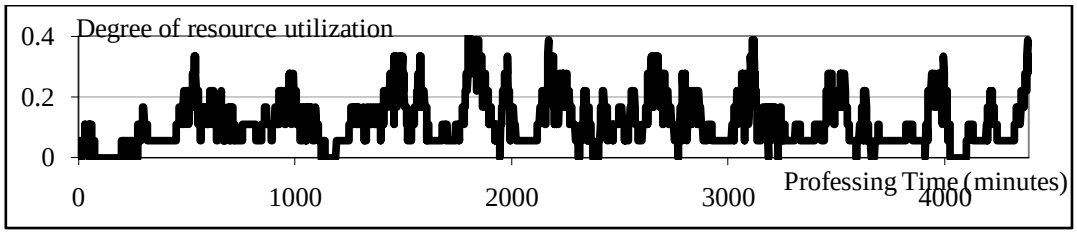


Figure VIII.12. Change of the degree of resource utilization during 100 remanufactured used products of the automotive part remanufacturing system.

- sv^{pp} range 1: 0.00 - 0.43;
- sv^{pp} range 2: 0.43 - 0.53;
- sv^{pp} range 3: 0.53 - 0.59;
- sv^{pp} range 4: 0.59 - 1.00.
- sv^{ru} range 1: 0.00 - 0.10;
- sv^{ru} range 2: 0.10 - 0.15;
- sv^{ru} range 3: 0.15 - 1.00.

Workstations should be classified into three groups: disassembly, refurbishment, and reassembly as discussed in section VIII.6.3.2. All workstations are definitely classified exclusively into one of the three groups except for the thermal cleaning workstation [WS02], because it handles both [WB] after disassembly and [AS] before disassembly. The calculated values of the belonging tendency $bt_{wg,w}$ (def. VIII.6.11) of [WS02] are 0.029 and 0.025 for the disassembly and refurbishment groups respectively, hence [WS02] is classified into the disassembly group; consequently workstations are grouped as follows:

- disassembly group: [WS01], [WS02], [WS13];

– refurbishment group: [WS03], [WS04], [WS05], [WS06], [WS07], [WS08], [WS09], [WS14], [WS15], [WS16];
– disassembly group: [WS11], [WS12], [WS17] , [WS18].

Table VIII.11 shows the selected dispatching rule for each system state variable range cases by comparing with the performance of the selected default dispatching rule: NEDD. Some cases like sv^{pp} range 1 and sv^{ru} range 3 have no better dispatching rule combinations than the default dispatching rule, hence NEDD is allocated to all workstations in such cases. Dispatching rules are not grouped and each dispatching rule is examined directly in the same way with the mobile phone repair system case, because the candidate dispatching rules are not so many.

7.2.1.3. Performance improvement analysis

Table VIII.12 represent the simulation results of two cases: the NEDD rule application case and the dynamic dispatching rule allocation and batch parameter application case. The tardiness related measures cannot be analyzed as eligible statistical information because of their high variance depending on simulation cases; for instance, the mean tardiness of 10 times simulation is distributed about from 200 to 4,000. Hence this thesis focus only on the flow time related measures. Applying the proposed scheduling mechanism improved the mean flow time by about 5.57%, but the t-test result shows the difference is not so definite; the difference confidence level is only about 80%. The maximum flow time and flow time STD show definite improvement of 38.66% and 30.99% respectively with over 99% confidence level.

In conclusion, the proposed scheduling mechanism improves the flow time measures, especially for the variance related measures: the maximum flow time and the flow time STD. The system states changes continuously during the remanufacturing, and the application of only one dispatching rule for the whole time can cause an unstable performance; for example, if a remanufacturing system usually is in busy states and a dispatching rule proper for the states are allocated to all workstations, the system performance can decrease in case the idle states are unexpectedly kept for a long time. The proposed scheduling mechanism monitors system states continuously and reacts instantly in case the allocated dispatching rule is not proper for the current system states. Hence it can overcome the weak point of the static dispatching rule allocation.

Table VIII.11. Best dispatching rule set for each system state of the automotive part remanufacturing system.

Selected dispatching rule (mean flow time (minutes))				
System state variables		Workstation group		
sv^{pp} range	sv^{ru} range	Disassembly	Refurbishing	Reassembly
Range 1	Range 1	SPT (566.4)	SPT (557.2)	
	Range 2	FCFS (541.2)	FCFS (564.2)	RPT (563.2)
	Range 3			
Range 2	Range 1		MDD (554.7)	FCFS (548.3)
	Range 2	EDD (570.5)		
	Range 3	FCFS (549.9)		EDD (572.5)
Range 3	Range 1			
	Range 2			SPT (545.3)
	Range 3			
Range 4	Range 1	EDD (551.1)		
	Range 2	WINQ (538.8)		
	Range 3		WINQ (573.5)	RPT (521.1)
Default dispatching rule: NEDD			575.6	

Table VIII.12. Performance of the automotive part remanufacturing system with/without the scheduling mechanism.

Model	Flow time (minutes)			Tardiness (minutes)			
	Mean	Maximum	STD	Mean	Maximum	STD	PoTJ[a] (%)
Scheduling mechanism (α)	643.0	2,235	411.4	1,440.1	11,297	2,412.2	0.313
One DR: NEDD (β)	680.9	3,644	596.1	1,700.7	12,139	2,721.8	0.343
Improvement[b] (%)	5.57	38.66	30.99	15.32	6.94	11.38	8.67
P-value from T-test	0.211	0.000	0.007	0.338	0.355	0.318	0.364

[a] PoTJ: Percentage of tardy jobs.
[b] [improvement] = $(\beta-\alpha)/\beta\times100$; the calculation results can be different with the value calculated with that in the table because of the value in the table is rounded from the original value.

Table VIII.13. Pilot simulation results to find the default dispatching rule for the example QRS.

Dispatching rule	Cost				Divided group
	Operation	Delay penalty	Disposal	Total	
WINQ	148.8	107.5	48.5	304.8	Group A
MDD	151.4	107.2	48.9	307.4	
EDD	152.8	113.2	49.8	315.7	Group B
LPOSU	148.1	118.9	48.8	315.8	
LPOS	148.9	122.4	47.6	319.0	
LPOST	149.5	119.8	50.6	319.9	
RPT	153.8	116.5	50.0	320.3	Group C
NEDD	154.3	116.1	50.5	320.9	
LPOSR	153.6	116.6	50.8	321.0	
LPOSD	151.3	120.9	50.1	322.3	
FCFS	152.6	120.6	51.5	324.6	
LPOSM	153.0	120.9	50.9	324.8	
SPT	155.7	125.4	52.1	333.2	exclusion

7.2.2. Effect in the QRS

This section validates the case of the application of all the proposed scheduling mechanism together: the PDSP's operation/resource selection, the batch processing scheduling, and the knowledge-based dynamic dispatching rule allocation. The example QRS contains the PDSP/resource quality information, hence it can apply the PDSP's operation/resource selection mechanism which was inapplicable in the conventional remanufacturing like the above two cases.

7.2.2.1. The example QRS (case study 3)

The example QRS contains quality information which can be utilized in the calculation formula of the proposed six dispatching rules in the above chapter VII: lowest probability of operation success (LPOS), LPOS to operation processing time (LPOST), LPOS to due date urgency (LPOSU), LPOS to remaining total processing time (LPOSR), LPOS to due date (LPOSD), and LPOS to modified due date (LPOSM). Hence this thesis selects all the proposed quality considered dispatching rules, the conventional benchmarking dispatching rules (refer to table VII.1), and the new earliest due date (NEDD) rule (Sakara 2006) which is developed for a remanufacturing system as candidates. The proposed total remanufacturing cost performance measure $\boldsymbol{M^{perf}}$ (def. VIII.2.1) is also applicable. Hence this section focuses on the system performance from the perspective of the total remanufacturing cost per used product. 3.58 is set for the due date adjustment factor of the TWK due date setting method.

Table VIII.13 represents the pilot simulation results to find the default dispatching rule. WINQ shows the best

performance in the total cost measure. Although there exist a better dispatching rule in the operation cost and delay penalty cost: LPOSU and MDD respectively, WINQ is the best overall. Hence this thesis selects the WINQ rule as the default dispatching rule for the example QRS.

7.2.2.1.1. Batch processing parameters and knowledge map

0.2 and 0.0 are selected for the batch processing parameters t^{sb}_r (def. VIII.5.2) and wt^{acf} (def. VIII.5.3) respectively in the same way with that applied for the automotive remanufacturing system. The total remanufacturing cost depending on t^{sb}_r with 0.5 of wt^{acf} is lowest around 0.1 and 0.2 t^{sb}_r (refer to figure VIII.13), and the pilot simulation results depending on wt^{acf} of 0.1 and 0.2 t^{sb}_r cases result in the 0.2 and 0.0 of t^{sb}_r and wt^{acf} combination shows the best performance (refer to table VIII.14).

The degree of resource utilization sv^{ru} (def. VIII.6.7) is arbitrarily selected as the system state variable for the knowledge map input variable, because the best performance dispatching rules are quite different depending on the resource utilization rate as discussed in section VII.2.3. Figure VIII.14 shows the change of sv^{ru} during the 100 remanufactured used products. The sv^{ru} range is divided into four ranges in the same way with the above two cases as follows:

- sv^{ru} range 1: 0 - 0.15;
- sv^{ru} range 2: 0.15 - 0.3;
- sv^{ru} range 3: 0.3 - 0.45;
- sv^{ru} range 4: 0.45 - 1.

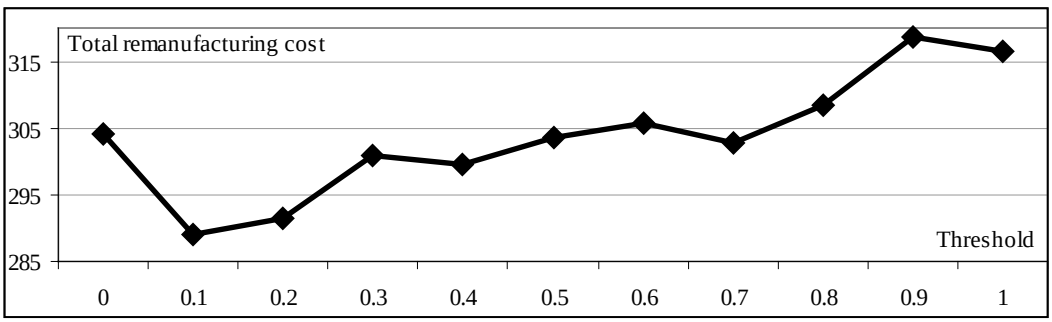


Figure VIII.13. Total remanufacturing cost per used product for each batch threshold with 0.5 of the average capacity fill up rate weight for the example QRS.

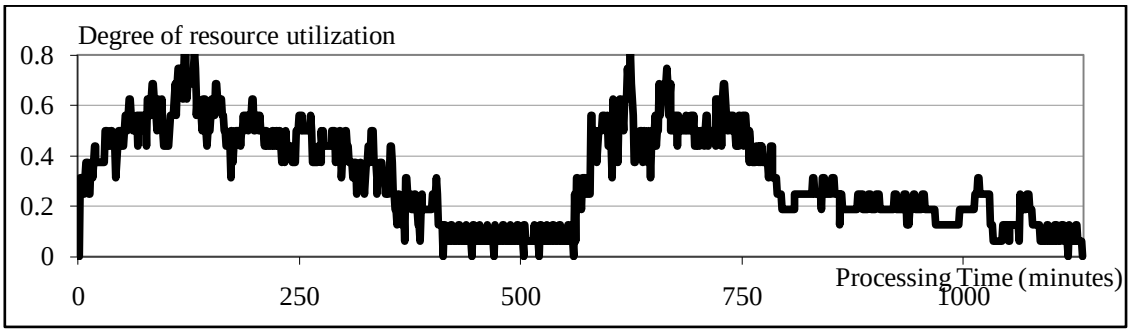


Figure VIII.14. Change of the degree of resource utilization during 100 used product handling of the example QRS.

Table VIII.14. Pilot simulation results to select the batch resource related parameters for the example QRS.

	Mean flow time (minutes)				
	Weight of the average capacity fill up rate				
Threshold	0.00	0.25	0.50	0.75	1.00
0.0	-	-	304.2	-	-
0.1	300.0	285.1	288.9	303.4	308.5
0.2	284.6	300.9	291.6	299.1	298.0
0.3	293.4	304.4	300.8	303.5	292.5
0.4	296.0	316.6	299.6	302.7	302.7

Table VIII.15. Selected best dispatching rule set for the system state variable ranges of the example QRS.

	Selected group/dispatching rule (total remanufacturing cost)	
sv^{ru} range	Workstation group	
	Disassembly	Refurbishment
Range 1	Group B (300.4) / LPOST (296.8)	-
Range 2	Group B (295.6) / LPOSU (293.8)	Group B (303.1) / LPOSU (282.9)
Range 3	-	Group A (297.5) / MDD

The workstation classification is not difficult, because their operations in charge are definitely classified. Hence the workstations are classified as follows:

– disassembly group: [WS1], [WS2], [WS3];
– refurbishment group: [WS4], [WS5], [WS6];
– reassembly group: [WS7], [WS8].

The number of candidate dispatching rules selected for the example QRS is not so small, hence this thesis classifies them into three groups which are found in table VIII.13. The SPT rule is excluded in the dynamic dispatching rule allocation mechanism because of its extraordinary bad performance. The group A includes only the MDD rule, because the WINQ rule is the default dispatching rule. Hence in case the group A is selected as the best dispatching rule group of a workstation group for a system state range, MDD is instantly selected as the best dispatching rule for the criteria without further simulations.

Table VIII.15 represents the pilot simulation results to find the best dispatching rule set for each system state range. No dispatching rule group shows the better performance than the default dispatching rule in the sv^{ru} range 4 and for the reassembly workstation group, hence they are not represented in the table. The dispatching rule group C never shows better performances.

7.2.2.1.2. Performance improvement analysis

Table VIII.16 shows the simulation results to verify the performance improvement by each scheduling mechanism and by the complete application of the proposed scheduling mechanism.

The batch processing mechanism reduces the total remanufacturing cost by 6.43% with over 99% confidence level. The operation cost has generally positive relation with operation processing time which also shows positive relation with delay time. But the operation cost decreases while the delay penalty cost increases by the mechanisms. Such a negative relation is inferred to be caused by the batch processing. The longer the waiting time before the batch process starts, the more PDSPs the batch resource processes simultaneously and the longer time the PDSPs flow the remanufacturing shop. Hence the selected batch processing parameters are inferred to be at the point, where the processing cost decrease effect mostly defeats the delay penalty cost increase effect.

Table VIII.16. Performance of the example QRS for each scheduling mechanism application case.

Scheduling mechanism	Cost				Comparison	
	Operation	Delay Penalty	Disposal	Total	Improvement[a] (%)	P-value
No mechanism (default)$^{(\alpha)}$	151.5	111.8	50.0	313.2		
Batch processing $^{(\beta)}$	119.1	124.6	49.4	293.1	6.43	0.000
Operation/resource selection $^{(\beta)}$	137.4	118.3	50.9	306.5	2.14	0.096
Dynamic dispatching rule $^{(\beta)}$	149.6	106.0	49.2	304.8	2.70	0.060
Whole mechanism $^{(\beta)}$	111.0	127.7	49.0	287.7	8.15	0.002

[a] [improvement] = (β-α)/β×100; the calculation results can be different with the value calculated with that in the table because of the value in the table is rounded from the original value.

The PDSP's dynamic operation/resource selection mechanism reduces the cost by 2.14% with about 90% confidence level. It shows the operation cost decreases, the delay penalty cost increases, and the disposal cost remains nearly the same; hence the PDSPs in the example QRS is inferred to give the higher value to the operation cost decrease effect by the more PDSPs simultaneous processing at the same time than the delay penalty cost increase effect by the longer waiting time. Disposal cost seems not to be considered because of its small portion in the total remanufacturing cost.

The dynamic dispatching rule allocation mechanism reduces the cost by 2.70% with about 94% confidence level. It reduces mainly the delay penalty cost because of its incapability to intervene the batch processing or operation/resource selection to affect the disposal or operation cost decrease. Although the mechanism can affect the processing time decrease, the operation processing cost in this case is inferred to be dominated rather by the batch processing.

The last row of table VIII.16 shows the definite remanufacturing cost reduction by 8.15% with over 99% confidence level in case all the three scheduling mechanisms are applied together. The 8.15% improvement is less than 11.27% (=6.43+2.14+2.70) which is the sum of improvement by each mechanism, hence combining the mechanisms does not result in any synergy effects. But it is greater than 6.43% which is the maximum improvement among exclusive mechanism application cases, that explains the combining the mechanisms does not cause a negative effect. Hence the application of all the mechanisms at the same time is more preferable than exclusive application of each mechanism. The reduction effect by each cost element is similar with that of the batch processing scheduling mechanism application case, hence it is inferred that the batch mechanism affects strongest the example QRS case. The effect of each mechanism can be different depending on the portion of each cost element. The applied batch processing parameters and the best dispatching rule knowledge are exclusively gathered without consideration of the other mechanisms, hence finding them with applying all the mechanisms in an integrated way can reduce the cost more.

IX. Conclusion and Further Research

This chapter summarizes the proposed quality embedded remanufacturing system (QRS) modeling tools, the quality embedded dispatching rules, and the knowledge-based scheduling mechanism with findings from simulation results. Related further research are also suggested last.

Abbreviated terms used in this chapter:

DTPN	Dynamic Token Petri-Net;
EWT	Estimated Waiting Time;
IRSR	Intuitive Remanufacturing System Representation;
LPOS	Lowest Probability of Operation Success;
LPOSR	LPOS to remaining total processing time;
LPOST	LPOS to operation processing Time;
LPOSU	LPOS to due date urgency;
PDSP	used Product and Disassembled Subassembly/Part;
POF	Probability of Operation Failure;
QRS	Quality embedded Remanufacturing System;
RPQ	Ratio of PDSP output Qualities after an operation
SPT	Shortest Processing Time;
STD	STandard Deviation.

1. Conclusion

1.1. Discussed topics

Remanufacturing is a good alternative from the perspective of the environment conservation, cost reduction, and so on. The remanufacturing system should be controlled in a different way with the conventional manufacturing systems because of its quite different characteristics, at the center of which is placed the uncertain quality of used products and disassembled subassemblies/parts (PDSPs). This thesis deals with the scheduling problem of a remanufacturing system with uncertain PDSP quality which is called the quality embedded remanufacturing system (QRS).

Whatever the objective is, the quality in the QRS should be clarified first. This thesis defines two kinds of qualities as follows:

- PDSP quality: the degree of difference between the obtained and required PDSP specifications;
- resource quality: the relative performance for an operation to the average performance of all resources which can process the operation.

The best remanufacturing process and resources to process the required operations are different depending on the change of PDSP quality as well as system states, and the proposed scheduling mechanism pursues performance improvement by reacting to those changes in real-time. Hence the important information for the proposed real-time scheduling mechanism is not the quality itself but the quality dependent operation processing statistics. This thesis classified operation processing results into four cases as follows (refer to figure IV.2):

- non-completion without additional defects;
- non-completion with additional defects;
- completion with additional defects;
- completion without additional defects (success case),

and defined a statistical information representation as follows:

$\boldsymbol{tf_{o,iq}}$ operation processing time statistics and probability of operation failure (POF) (def. IV.2.2);

$\boldsymbol{RPQ^{*}_{o,iq}}$ ratio of PDSP output qualities after an operation (RPQ) for the above four cases (def. IV.2.9, IV.2.11 – IV.2.13);

$\boldsymbol{rq_{o,r}}$ resource quality correction factor for the operation processing time and POF (def. IV.2.16).

The quality of used products at arrival time is uncertain and the quality cannot be exactly examined before disassembly. The quality of PDSPs even changes dynamically because of errors during disassembly or refurbishing operations. Hence individual PDSP controls and dynamic remanufacturing process change depending on its changing quality is preferable. This thesis suggests adopting an agent-based approach to deal with it; defining each PDSP as an autonomous agent makes it efficient to control each PDSP differently in its own way. The proposed multi-agent framework comprises four agent groups:

– PDSP agent group;
– facility agent group;
– PDSP life management agent group;
– knowledge support agent group.

Each element in the real-world remanufacturing system is defined as an agent which belongs to the PDSP and facility agent groups, and the PDSP agents are created by the agents in the PDSP life management group depending on the used product arrival and PDSP disassembly/reassembly. This thesis adopts a knowledge-based approach to overcome the information lagging problem originated from the computation time of real-time simulations, and the knowledge support agent group deals with it. The multi-agent framework also includes the communication protocol to be followed by agents; it is composed of two layers: the syntax and communication sequence layers.

A QRS modeling tool is required to create a multi-agent system corresponding to a real-world remanufacturing system. This thesis defined two QRS modeling tools as follows because of no eligible existing tools:

– user-side modeling tool: intuitive remanufacturing system representation (IRSR);
– system-side modeling tool: dynamic token Petri-nets (DTPN).

IRSR encloses all the required information for the QRS control and simulation. DTPN is extended Petri-nets capable of dynamic token controls based on the statistical information and token's internal attributes. A conversion method from an IRSR model to a DTPN model is also proposed to maintain the consistency of two models. The virtual multi-agent system is constructed based on the IRSR model and each created agent contains corresponding DTPN modules.

The simplest way of performance improvement is to apply proper dispatching rules, hence this thesis proposed six dispatching rules considering the PDSP/resource quality. The PDSP priority calculation function of the proposed lowest probability of operation success (LPOS) rule is as follows:

$$\boldsymbol{pri}^{LPOS}_{r,o,iq} = \mathbf{1} - \boldsymbol{pof}_{o,iq} \times \boldsymbol{cf}^{\,r-POF}_{\;o,r}$$ (def. VII.1.1).

LPOS gives the highest priority to the PDSP requesting an operation which is most expected not to succeed. It also involves the resource quality factor $\boldsymbol{cf}^{r\text{-}POF}_{o,r}$ (def. IV.2.18) by multiplying with the POF $\boldsymbol{pof}_{o,iq}$ (def. IV.2.7). The other five dispatching rules are derived based on LPOS by combining it with popular conventional benchmark dispatching rules; for example, the LPOS to operation processing time (LPOST) rule is the combination of LPOS and the shortest processing time (SPT) rule.

This thesis tries to improve the integrated overall performance of the QRS, and an integrated performance measure is required to cope with it. Hence a cost based performance measure is proposed: the total remanufacturing cost per used product $\boldsymbol{M}^{perf}$ (def. VIII.2.1) which encloses the cost for the operation processing, PDSP disposal, and delay penalty.

The proposed scheduling mechanism is mainly composed of the following two:

– PDSP's dynamic operation/resource selection depending on its quality, resource quality, and the estimated waiting time (EWT) for operation processing;
– dynamic dispatching rule allocation to workstations depending on the system states based on the knowledge accumulated by off-life simulations

The remanufacturing system contains batch resources and consequently additionally applies the batch processing scheduling method which this thesis does not handle intensively; the batch processing starts when the waiting queue state variable s^{wq}_r (def. VIII.5.1) exceeds the preset threshold t^{sb}_r (def. VIII.5.2). The proper t^{sb}_r and other required variable for s^{wq}_r calculation are selected by simulations.

The PDSP selects an operation and a resource by comparing the estimated performance $M^{m\text{-}perf}$ (def. VIII.4.7) for possible operation/resource selection cases. The selection is consecutively done after every operation completion and is affected by the quality and EWT of capable resources as well as its current quality, hence the remanufacturing process dynamically changes and is different from PDSP to PDSP. An indirect calculation method by a recursive manner is proposed (refer to figure VIII.5) because of the difficulty of the straightforward $M^{m\text{-}perf}$ calculation.

The PDSP's operation/resource selection concentrates only on its private performance; in other words, the mechanism pursues local performance maximization. Hence the dynamic dispatching rule allocation method is supported for the whole system performance maximization. In the proposed multi-agent framework exists the dispatching rule allocator agent which allocates the best dispatching rule to each workstation depending on the current system states based on the mounted knowledge. The knowledge is structured in a decision tree form which gets system state variables as an input and returns a dispatching rule allocation scenario. The gathered scenario can be converted into the best dispatching rule set for workstations by the scenario map of a matrix form which is also mounted in the dispatching rule allocator agent.

The knowledge is accumulated by off-line simulations by the simulator agent. The simulator agent collects the statistical information on the operation processing, simulates for possible system state cases, analyzes simulation results, and creates knowledge. The created knowledge is mounted into the dispatching rule allocator agent without system down time. This thesis proposed a heuristic method to reduce simulation complexity, because the off-line simulation also requires tremendous computation time; too many combinations of workstations, dispatching rules, and system state values are possible. The proposed heuristic method reduces computation time of each system state value case from $O(n^m)$ of full enumeration case to $O(n)$ by classifying workstation into three groups and selecting the best dispatching rule for a workstation group in two steps:

– select the best dispatching rule group for a workstation group;
– select the best dispatching rule for a workstation group among the dispatching rules that belong to the selected dispatching rule group.

Applying the above discussed approaches and mechanisms enable a remanufacturing system not only to be

controlled with involving its uncertain quality characteristics but also to pursue its performance maximization.

1.2. Benefits and findings

The proposed modeling tools can properly model the QRS. Its user-side model increases the understandability of field technicians, consequently the model representing a target remanufacturing system can be easily validated; that was difficult with Petri-nets or network type models. Although tools are developed to handle the quality information of the QRS, it can also be utilized without quality modeling. This thesis showed the IRSR's capability by emulating existing remanufacturing system modeling which have no quality related features.

Some of the proposed six dispatching rules showed performance improvement compared to the popular benchmark dispatching rules with the example QRS in chapter III from the perspective of the conventional performance measures: the flow time and tardiness measures (refer to table VII.2). The LPOS to due date urgency (LPOSU) rule in a busy system showed 2.45% and 5.15% better performances in the mean flow time and tardiness respectively than the modified due date (MDD) rule which was the best among the conventional popular benchmark rules. The LPOS to remaining total processing time (LPOSR) rule in an idle system showed 6.26% and 4.67% better performances in the same measures than the remaining processing time (RPT) rule. But the Student t-test results indicated that the improvement is not definite; the improvement by LPOSU is statistically confident only with about 70% level, and that by LPOSR is with about 90% and 75% levels for the mean flow time and tardiness respectively. On the contrary, LPOSR in an idle system showed definite improvement of the variance related measures as expected at the dispatching rule design time; the standard deviation (STD) of the flow time and tardiness is improved 11.33% and 12.50% respectively with about 95% confidence level. The percentage of tardy jobs measure is rather aggravated; LPOSR aggravated the performance by 2.94% with about 94% confidence level. The difference of the best dispatching rule depending on the system states results in the necessity of the dynamic dispatching rule allocation depending on system state.

This thesis showed the proposed dynamic dispatching rule allocation mechanism is effective to the remanufacturing systems (refer to tables VIII.5 and VIII.12). The mechanism was applied to a mobile phone repair system without any additional scheduling mechanisms for batch processing scheduling and PDSP's dynamic remanufacturing process selection; it showed definite performance improvement by 4.01% and 8.31% with about 95% confidence level in the mean flow time and tardiness measure respectively, and also definitely improved 1.74% with 99% confidence level in the percentage of tardy job measure. The dynamic dispatching rule allocation mechanism did not definitely improve the mean flow time performance in the other case: an automotive part remanufacturing system; the performance is improved 5.57%, but the confidence level of it was under 80%. On the contrary, it definitely improved the variance related performance; the maximum and STD of the flow time are improved 38.66% and 30.99% respectively in over 99% confidence level.

The remanufacturing cost decrease effect by the whole scheduling mechanism application and the marginal effect by each mechanism application was also validated with the example QRS (refer to table VIII.16). The batch processing and dynamic dispatching rule allocation mechanisms definitely reduced the total remanufacturing cost by 6.43% and 2.70% with over 99% and 94% confidence level respectively, and the PDSP's dynamic operation/resource selection mechanism also reduced 2.14% with about 90% confidence level.

The remanufacturing cost difference of the example QRS is inferred to be dominated by the batch processing cost; total cost reduction is affected mainly by the operation cost reduction. The whole mechanism application reduced 8.15% of the total cost with over 99% confidence level. Combining all three mechanisms caused no definite synergy effects in the example QRS case, but it did not cause any negative effects neither. Hence it is preferable to apply all the scheduling mechanism in an integrated way.

The proposed scheduling mechanism showed a reasonable performance improvement in the three examined cases, and the better performance is expected by the following actions:

- enclose additional system state variables;
- full enumeration for the best dispatching rule set selection;
- more sophisticated pilot simulations for batch parameter selection;
- integrated consideration of the three scheduling mechanisms: the batch processing scheduling, PDSP's dynamic operation/resource selection, and dynamic dispatching rule allocation mechanisms.

2. Further related research

Although this thesis proposed a multi-agent framework, modeling tools, and real-time scheduling mechanism for the QRS and validated their benefits, some related issues to be researched are still remained. The proposed mechanisms can be enhanced by further research from the perspective of the system performance improvement, simulation time reduction, and so on. Hence the last this section suggests some topics for the further related research as follows:

- representation method of PDSP quality itself;
- mathematical integration of processing time statistics depending on input PDSP qualities;
- sophisticated representation of the resource quality;
- straightforward calculation method to estimate the proposed cost-based performance measures;
- waiting time estimation method;
- batch processing mechanism considering the specific characteristics of the remanufacturing system;
- systemic approach for the system state variable range division;
- heuristics for better knowledge generation in shorter simulation time.

X. Appendices

Appendix A. Basic symbols for mathematical expression

$\mathbb{B}$ set of boolean values, element of which are *true* and *false*;

$\mathbb{C}$ set of cost values;

$\mathbb{I}^{+}$ set of positive integers;

$\mathbb{I}^{0+}$ set of non-negative integers, i.e., $\mathbb{I}^{+} \cup \{0\}$;

$\mathbb{R}$ set of real numbers;

$\mathbb{R}^{+}$ set of positive real numbers;

$\mathbb{R}^{0+}$ set of non-negative real numbers, i.e., $\mathbb{R}^{+} \cup \{0\}$;

$\mathbb{T}$ set of time: durations or time instance;

$\mathbf{F}^{\mathbf{PDF}} = \{(dt, parm_1, parm_2, \ldots) \mid dt \in \mathbf{DT}$ and $parm_i \in \mathbb{R}\}$
set of probability density functions (PDFs);

inf. infinite number;

avg(***NoS***) function returns the average value of the elements in ***NoS***;

var(***NoS***) function returns the variance value of the elements in ***NoS***;

max(***NoS***) function returns the maximum value of the elements in ***NoS***;

min(***NoS***) function returns the minimum value of the elements in ***NoS***,

where

DT set of distribution type, elements of which are deterministic, uniform, normal, exponential, and so on;

$parm_i$ required parameters for the distribution ***dt***, for example, the normal distribution needs two parameters of the average and standard deviation.

NoS number set or series of numbers.

Here, the function ***avg*** can be applied with a different form as follows:

$$\underset{e \in NoS}{avg}\left(f(e) \right) = avg(\{f(e) \mid e \in NoS\}).$$

The other three functions, ***var***, ***max***, and ***min***, also can have the form above.

Appendix B. Remanufacturing cost calculation of the example QRS

The processing cost of each operation is found in table B.1. Each cost is calculated by multiplying the average processing time and the processing cost per unit time without consideration of reworks for simplicity, except for the batch operations by the workstations [WS4] and [WS5]. The processing cost by batch resources should be equally distributed to the simultaneously processed PDSPs, the number of which is different from time to time. This thesis simply divides the processing cost by the average lot size of each used product arrival under a simple assumption; all parts disassembled from the used products in the same bulk arrival are processed at the same time in a batch resource; for example, the processing cost of the operation OPa06 is 1.98 (= 16.2×3/24.5) which is calculated by dividing the processing cost 48.6 (=16.2×3) by the average lot size 24.5 of the used product [A].

The remanufacturing cost of a used product is the sum of the processing cost of all operations. While the processing cost of cleaning or other operations are the same with the sum of the processing cost for all the corresponding operations, the disassembly cost is not a simple sum of all disassembly operation processing costs, because some operations are alternatively processed; for example, the operation OPa01 is not processed in case the operation OPa02 is processed. Hence this thesis calculates the disassembly cost as the average value of the processing costs of the alternative disassembly processes; for example, [A] can be disassembled by the operations OPa01 and OPa04 or by the operations OPa02 and OPa03, hence the disassembly cost of [A] is 37.54 (= ((19.20+24.48)+(18.60+12.80))/2) (refer to the operation processing cost of the operations OPa01 - OPa04 in table B.1). The second columns of the second last rows of tables B.2 and B.3 show the delay penalty costs of the used products [A] and [B] respectively, which are derived from their remanufacturing costs.

Table B.1. Processing cost of each operation in the example QRS.

Used product		Operation		Workstation	average	processing cost	Processing
ID	Lot size	ID	Type	in charge	processing time	per unit time[a]	cost
[A]	24.5	OPa01	disassembly	WS2	9.6	2	19.20
		OPa02	disassembly	WS2	9.3	2	18.60
		OPa03	disassembly	WS2	6.4	2	12.80
		OPa04	disassembly	WS2, WS3	8.9	2.75	24.48
		OPa05	cleaning	WS6	6.4	3	19.20
		OPa06	cleaning	WS5	16.2	3	1.98
		OPa07	machining	WS4	4.4	2.5	0.45
		OPa08	reassembly	WS7	8.7	2	17.40
		OPa09	test	WS8	2.8	1	2.80
[B]	22.4	OPb01	disassembly	WS1	2.3	1	2.30
		OPb02	disassembly	WS1	12.6	1	12.60
		OPb03	disassembly	WS2, WS3	6.1	2.75	16.78
		OPb04	disassembly	WS1	10.3	1	10.30
		OPb05	disassembly	WS2, WS3	6.9	2.75	18.98
		OPb06	disassembly	WS2	1.9	2	3.80
		OPb07	cleaning	WS5	30.4	3	4.07
		OPb08	cleaning	WS6	20.8	3	62.40
		OPb09	machining	WS4	9.7	2.5	1.08
		OPb10	machining	WS4	8.6	2.5	0.96
		OPb11	cleaning	WS6	18.9	3	56.70
		OPb12	cleaning	WS5	13.7	3	1.83
		OPb13	cleaning	WS5	18.8	3	2.52
		OPb14	reassembly	WS7	11.5	2	23.00
		OPb15	test	WS8	3.1	1	3.10

[a] 2.75 (=(2.5+3)/2) in the processing cost per unit time is the average value of the disassembly cost per unit time by human resources (WS2) and the disassembly cost per unit time with specific tools (WS3).

Table B.2. Retreatment cost of the used product [A] and its composing part in the example QRS.

Operation			Retreatment cost			
ID	Type	processing cost	[A]	[A1]	[A2]	[A3]
OPa01 - OPa04	disassembly	37.54	37.54	12.51	12.51	12.51
OPa05	cleaning	19.20	19.20	19.20		
OPa06	cleaning	1.98	1.98		1.98	
OPa07	machining	0.45	0.45		0.45	
OPa08	reassembly	17.40	17.40			
OPa09	test	2.80	2.80			
PDSP remanufacturing cost[a]			79.37	31.71	14.95	12.51
Delay penalty base cost[b]			40			
Part disposal cost				95	45	38

[a] Sum of the processing cost of related operations. The value can be a little different from the value calculated based on the values in the table because of rounding.
[b] The delay penalty cost is about half of the remanufacturing cost and the disposal cost is 3 times of the retreatment cost of each part.

Table B.3. Retreatment cost of the used product [B] and its composing part in the example QRS.

Operation			Retreatment cost					
ID	Type	processing cost	[B]	[B1]	[B2]	[B3]	[B4]	[B5]
OPb01 - OPb06	disassembly	35.43	35.43	7.09	7.09	7.09	7.09	7.09
OPb07	cleaning	4.07	4.07	4.07				
OPb08	cleaning	62.40	62.40		62.40			
OPb09	machining	1.08	1.08		1.08			
OPb10	machining	0.96	0.96			0.96		
OPb11	cleaning	56.70	56.70			56.70		
OPb12	cleaning	1.83	1.83				1.83	
OPb13	cleaning	2.52	2.52					2.52
OPb14	reassembly	23.00	23.00					
OPb15	test	3.10	3.10					
PDSP remanufacturing cost[a]			191.09	11.16	70.57	64.74	8.92	9.60
Delay penalty base cost[b]			100					
Part disposal cost				33	212	194	27	29

[a] Sum of the processing cost of related operations. The value can be a little different from the value calculated based on the values in the table because of rounding.
[b] The delay penalty cost is about half of the remanufacturing cost and the disposal cost is 3 times of the retreatment cost of each part.

The cost of a part, replaced by disposal and substitution, is set as 3 times bigger than the sum of its disassembly, cleaning, and refurbishing costs. The cleaning and refurbishment cost of a part is the processing cost of its cleaning and refurbishment operations; for example, the cleaning cost of the part [A1] is 19.20 which is the processing cost of the operation OPa05 (refer to figures III.2 and III.3). While, the disassembly operation processing cost of a part is not easily gatherable, because a disassembly operation involves two or more parts simultaneously. This thesis equally distributes the whole disassembly cost of a used product to the composing parts of the used product; for example, the disassembly cost of the part [A1] of the used product [A] is 12.51, which is one third of [A]'s disassembly cost 37.54 (refer to the first row of table B.2). The last rows of tables B.2 and B.3 shows the disposal cost of each part of [A] and [B] respectively.

Appendix C. Parameterization of PDSP quality

For the conclusive objective of this thesis, in other words, to propose a scheduling mechanism, a different approach with conventional manufacturing systems is required in the quality specifications for PDSPs. In QRS, the functional specifications of products and subassemblies need no more to be considered, because their functions vanish away by the disassembly operation. Instead, the following should be mainly considered:

– the effect of joint defects among subassemblies and parts on the disassembly operation;
– the recoverability of defective parts by retreatment process like machining or cleaning.

The disassembly operation time and quality is highly dependent on the condition of joints among subassemblies and parts to be disassembled. The quality of composing parts could also affect the disassembly operation; for example, a subassembly cannot be fixed in a stable manner due to its damage, and this can disturb the disassembly operation. But it is not so frequent and has too diverse cases to be systematically synthesized. Therefore this thesis focuses on the effect of joint defects on the disassembly operation.

The condition of a joint can be different depending on the types of joints. The subassemblies and parts can be assembled by the following three types of joints:

Type 1: non-permanent joining with joining elements; for example, assembly with bolts and nuts;
Type 2: non-permanent joining by tight matching without joining elements; for example, joining by a press-fitting method;
Type 3: permanent joining; for example, assembly by welding and joining with soldering.

In type 1 joining, a defect of a joining element by rust, abrasion, and so on negatively affects the disassembly time or quality of disassembled subassemblies and parts. On the contrary, the loosened joining elements can affect positively both of them. Disassembly of the type 2 joints is affected negatively by a defect of parts in the contact interface by physical transformation, rust, chemical adherence, and so on. Type 3 joining is not usually disassembled in the remanufacturing domain, hence those joints are not considered in this thesis (refer to assumption. I.3). The possible defects and their effects are different from subassembly to subassembly, hence this thesis suggests only the overall classification depending on the effects on the disassembly operation time and quality in the consecutive sections.

Before refurbishment, disassembled parts which are irrecoverable should be filtered out by quality examination of the disassembly performance. Even though disassembled parts may be recoverable, the retreatment time and the quality of retreated parts can be affected by the quality of input disassembled parts.

C.1. Parameterization of part quality

The quality $\boldsymbol{q_{cp}}$ of a composing part $\boldsymbol{cp}$ of a used product, can be defined as follows:

$q_{cp} \in PQ_{cp} = \{UIQ, ND, D^{nr}_{cp,1}, D^{nr}_{cp,2}, \ldots, D^{r}_{cp,1}, D^{r}_{cp,2}, \ldots\}$;

where

PQ_{cp} set of possible qualities of ***cp***;
UIQ unidentified quality;
ND no defect;
$D^{nr}_{cp,a}$ type ***a*** non-recoverable defect which can occur to ***cp***;
$D^{r}_{cp,b}$ type ***b*** recoverable defect which can occur to ***cp***.

Usually, quality of hidden parts inside a product or a subassembly cannot be identified, and ***UIQ*** corresponds to such case. Quality of such products is examined after some disassembly operations. ***ND*** means that the part meets the required specifications for reassembly, therefore the part does not need any more retreatment or function repair operations like machining, cleaning, defective chips replacement on a printed circuit board, and so on. $D^{nr}_{cp,a}$ means the part has fatal defects. Hence any retreatment operations are meaningless, and the part should be disposed. $D^{r}_{cp,b}$ represent classified defect types from physical and functional perspectives. The number of defect types is different from product to product, and each of them should have additional information about which retreatment operation is required to obtain the quality ***ND***.

C.2. Parameterization of product and subassembly quality

The quality of products or subassemblies has two categories: quality of joints and quality of composing parts. But there are multiple elements in each categories; in other words, a subassembly may have multiple joints and multiple parts. The quality of composing parts is just an aggregation of quality information of each composing parts. Therefore it can be simply represented by listing each part's quality. While the quality of a part is just for the representation of the isolated states in the part itself, the quality of a joint is for the parts connection state. It means that each joint should be identified with consideration of the connected parts by the joint. Therefore this section discusses how to identify joints, before discussing the representation of subassembly quality.

A subassembly can have many joints, and it can be considered that each joint connects two parts. Sometimes a joint can be shared by three or more parts; for example, a bolt and a nut can fix three parts at one time. But this thesis decomposes the joining structure step by step; two parts are unstably joined intermediately, and the aligned two parts and the remaining one are assembled with the bolt and the nut subsequently (refer to figure. C.1). Although this thesis differentiates such joints as multiple steps for the parameterization, the operation of disassembling such joints can be described as just one operation for the conciseness and for reflecting reality.

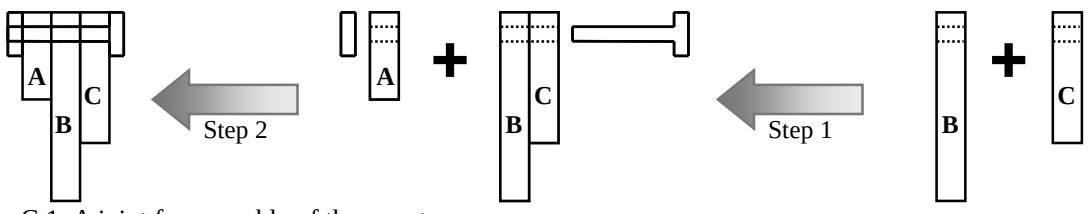


Figure C.1. A joint for assembly of three parts.

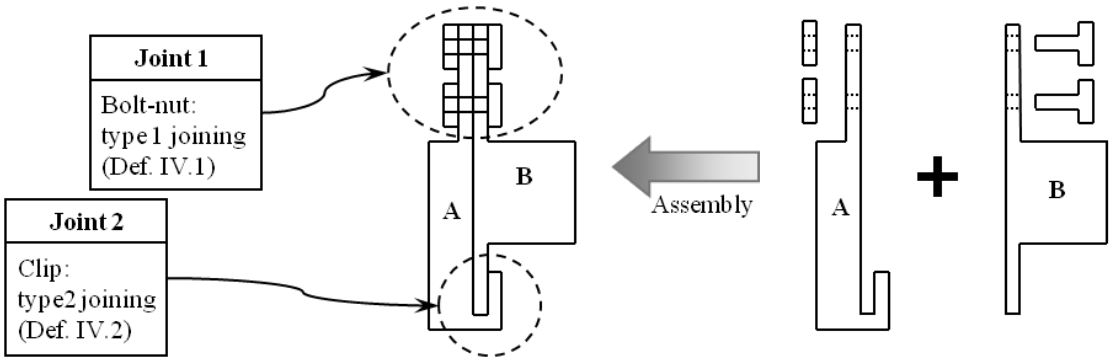


Figure C.2. Two joints of different joining types.

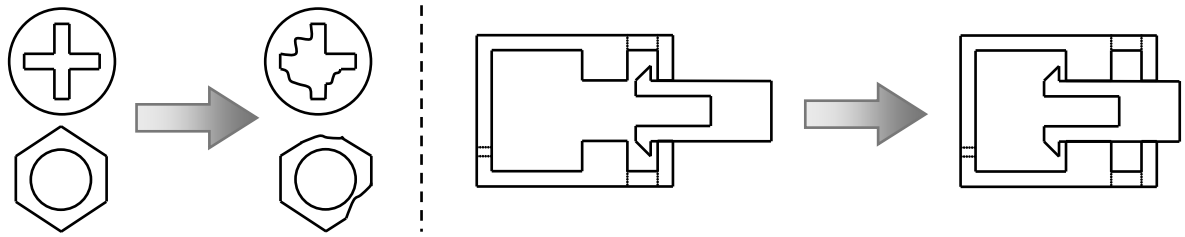


Figure C.3. Example of joint defects: (a-1) non-defective bolt and nut, (a-2) defective bolt and nut, (b-1) correctly aligned two parts, (b-2) misaligned two parts.

On the contrary, two parts can be joined by two or more joints, for example two parts are joined with two joints, one of which is by clip type and the other is with a bolt and a nut. But this thesis considers multiple same type joining elements as one joint, even in the case that the joining elements are not concentrated at the same spot. Because it is not a different joining method but just for strength and stability of joining (refer to figure C.2).

Each joint can have many kinds of defects. The type 1 joint can be damaged by rust, wear, adherence of joining elements, and so on. Figure C.3(a) shows an example of wear defects which can occur on bolts and nuts. Such defects usually occur in isolation in the joining elements themselves, but some defects like adherence can occur among the joining elements and the parts. Possible defects of the type 2 joint are adherence, cracks, misalignment (refer to the figure C.3(b)), and so on.

But as discussed at chapter IV, the important characteristics of joints are not the possible defects themselves or relative disassemble-ability of a joint to other joints but the effect of defects on the operation processing time and the POS during the disassembly of the subassembly. Hence this thesis represents the quality characteristics of joints from the perspective of the effect on the disassembly operations.

The failure of disassembly operation is divided into two kinds as follows:

– separation failure of connected subassemblies or parts;
– additional defects occurrence to parts or joining elements during disassembly.

In conclusion, this thesis deals with three kinds of information for each joint as follows:

– the effect on disassembly processing time;
– the effect on separation failure of connected subassemblies or parts;
– the effect on additional defects occurrence to parts and joining elements.

This thesis uses the term of 'additional defects', because the input product or subassembly can already have defects before the operation. As discussed above, the effect of defects on the disassembly operations is not always negative. In other words, some defects can decrease the processing time or increase the POS of the disassembly operation like a loose connection. But it should be also considered that the effect on the disassembly processing time, separation failure, and additional defects occurrence is not always in the same direction. Some defects can decrease the disassembly processing time as well as the POS; for instance, when a joining element is adhered to one of the connected parts by the joining element and it is impossible to disassemble without breaking the parts, the joint should be disassembled in a destructive manner which usually needs less time than normal disassembly. Therefore such defects can be considered as having a positive effect on disassembly time and negative effect on additional defects occurrence. Consequently the three kinds of effects should be handled individually, and each effect also should be classified into positive or negative effect.

As discussed above, a joint in a subassembly can be identified as a joint between two parts. This thesis represents joint quality information in a matrix form, rows and columns of which are list of composing parts. Each cell may have multiple values respectively, because the parts can be joined by two or more joints.

The quality $\boldsymbol{q_s}$ of a subassembly $\boldsymbol{s}$, which is composed of $\boldsymbol{n}$ parts, can be defined by the combination of composing parts and joints qualities, $\boldsymbol{q^{CP}_s}$ and $\boldsymbol{q^{J}_s}$, in the following form:

$\boldsymbol{q_s} = (\boldsymbol{q^{CP}_s}, \boldsymbol{q^{J}_s})$,
$\boldsymbol{q^{CP}_s} = (\boldsymbol{q_{cp1}}, \boldsymbol{q_{cp2}}, \ldots, \boldsymbol{q_{cpn}})$, and
$\boldsymbol{q^{J}_s} := (\boldsymbol{q^{J}_{s,i,j}})_{i<j,\ i,j = 1, 2, \ldots, n}$,

where

$\boldsymbol{q_{cpi}}$ the quality of composing part $\boldsymbol{cp_i}$ of a subassembly $\boldsymbol{s}$.

Here the $\boldsymbol{q^{J}_s}$ is an $\boldsymbol{n}$ by $\boldsymbol{n}$ matrix, the element of which contains the quality information of the joints of composing parts $\boldsymbol{cp_i}$ and $\boldsymbol{cp_j}$ which are joined by $\boldsymbol{m}$ joints, $\boldsymbol{q^{J}_{s,i,j}}$, and it can be represented as follows:

$\boldsymbol{q^{J}_{s,i,j}} = (\boldsymbol{q^{J}_{s,i,j,1}}, \boldsymbol{q^{J}_{s,i,j,2}}, \ldots, \boldsymbol{q^{J}_{s,i,j,m}})$,
$\boldsymbol{q^{J}_{s,i,j,k}} = (\boldsymbol{q^{J,TE}_{s,i,j,k}}, \boldsymbol{q^{J,FE}_{s,i,j,k}}, \boldsymbol{q^{J,DE}_{s,i,j,k}})$, and
$\boldsymbol{q^{J,TE}_{s,i,j,k}} \in \boldsymbol{PJQ^{TE}_{s,i,j,k}} = \{\boldsymbol{UITQ}, \boldsymbol{NTQ}, \boldsymbol{TQ^{P}_1}, \boldsymbol{TQ^{P}_2}, \ldots, \boldsymbol{TQ^{N}_1}, \boldsymbol{TQ^{N}_2}, \ldots\}$,
$\boldsymbol{q^{J,SE}_{s,i,j,k}} \in \boldsymbol{PJQ^{SE}_{s,i,j,k}} = \{\boldsymbol{UISQ}, \boldsymbol{NSQ}, \boldsymbol{SQ^{P}_1}, \boldsymbol{SQ^{P}_2}, \ldots, \boldsymbol{SQ^{N}_1}, \boldsymbol{SQ^{N}_2}, \ldots\}$,
$\boldsymbol{q^{J,DE}_{s,i,j,k}} \in \boldsymbol{PJQ^{DE}_{s,i,j,k}} = \{\boldsymbol{UIDQ}, \boldsymbol{NDQ}, \boldsymbol{DQ^{P}_1}, \boldsymbol{DQ^{P}_2}, \ldots, \boldsymbol{DQ^{N}_1}, \boldsymbol{DQ^{N}_2}, \ldots\}$,

where

$k = \{1, 2, \ldots, m\}$	sequential index for joints between parts cp_i and cp_j;
$q^J_{s,i,j,k}$	quality of joint k between a part p_i and a part p_j in a subassembly s;
$q^{J,TE}_{s,i,j,k}$	quality of joint k from the perspective of the processing time effect;
$q^{J,SE}_{s,i,j,k}$	quality of joint k from the perspective of the separation failure effect;
$q^{J,DE}_{s,i,j,k}$	quality of joint k from the perspective of the additional defects occurrence effect;
$PJQ^{TE}_{s,i,j,k}$	set of possible joint qualities of $q^{J,TE}_{s,i,j,k}$;
$PJQ^{SE}_{s,i,j,k}$	set of possible joint qualities of $q^{J,SE}_{s,i,j,k}$;
$PJQ^{DE}_{s,i,j,k}$	set of possible joint qualities of $q^{J,DE}_{s,i,j,k}$;
$UI*Q$	unidentified quality;
$N*Q$	quality having neutral effect, i.e. no effect on the processing time or failure of a disassembly operation;
$*Q^P_a$	type a quality having positive effect;
$*Q^N_b$	type b quality having negative effect.

Each type of positive and negative effects should be defined with consideration of possible defects of joints and their effects.

C.3. Simplification of quality instance representation

The quality of a subassembly can be expressed as a combination of one of possible joint quality of each joint and one of possible physical and functional quality of each composing part respectively. Therefore the greater number of composing parts in a subassembly is, the greater possible quality states of the subassembly are possible. But the number of possible quality states is too big to specify the statistical information for each one. To resolve this problem, this thesis adopts a set of frequently appearing qualities FQ_p of a PDSP p with the mapping function f^{dq}_p which specifies the corresponding detailed quality of elements in a FQ_p as follows:

$FQ_p = \{FAQ^p_1, FAQ^p_2, \ldots\}$	a set of frequently appearing qualities of a PDSP p;
$f^{dq}_p(q_p) \in FQ_p$	a mapping function specifying the detailed quality of a PDSP p,

where

FAQ^p_i	type i frequently appearing quality of a PDSP p;
q_p	quality of a PDSP p; in other words, q_{cp} in case of the p is a part or q_s in case of the p is a product or a subassembly.

With this approach we can simplify the definition of a PDSP quality q_p, which is defined in chapter IV.

Appendix D. Definition of XCPN (Sakara 2006)

This section represents the formal definition of extended two-level colored Petri-nets (XCPN), which is extracted from the documents containing the original definition (Sakara 2006).

D.1. Basic notations for the definition

The list of semantics and notations which used in the formal definition and commonly accepted in the colored Petri-nets definition is given below. Notions are presented in alphabetical order.

A / A_j	set of arcs of system net / token net;
AA	set of additional arcs;
B(t)	set of all bindings of transition t;
$b(v \in V) \in Type(v)$	*binding of a variable* v;
BE	set of all binding elements;
BES	set of all synchronized binding elements;
C	color function;
$CP = \bigcup_{s \in S} CP_s$	set of communication ports;
CP_s	set of communication ports of module M_s;
E / E_j	arcs expression function of system net / token net;
expr<*b*>	*the value obtained by evaluating an expression*, expr, *in a binding*, b;
I / I_j	initialization function of system net / token net;
j	index for token nets;
M / M_j	marking of a system net / token net;
M(p)	marking function that gives the number of tokens in the place p for a token net;
M(p, c)	marking function that gives the number of tokens of the color c in the place p for a system net;
M_0 / M_{j0}	initial marking of a system net / token net;
MI_s	interface of the module s;
N	node function;
NT(n)	node type function, that determines if n is a place or a transition;
P / P_j	set of places of system net / token nets;
PF	set of place fusion sets;
PT(p)	port type function that defines if port p is the input or output port;
S	set of all modules;
SN	system Net;
SP	set of synchronization ports;
$SPA = \bigcup_{s \in S} SPA_s$	synchronization ports assignment function;
SPA_s	the synchronization port assignment function for module M_s;

SSP	set of synchronized synchronization ports;
SY	set of synchronized transition steps;
T / T_j	set of transitions of system net / token nets;
TE	set of al token elements;
TF	set of transitions fusion sets;
TN	set of token nets;
Type(expr)	type of an expression expr;
Type(v∈V)	type of variable v;
Var(expr)	*set of variables in an expression* expr;
$Var(t) = \{v \mid \exists a \in A(t): v \in Var(E(a))\}$	set of variables associated with transition t;
Y	step;
YS	synchronized step.

D.2. Formal definition of XCPN

D.2.1. Time in the extended two-level colored Petri-net (XCPN)

The time in the XCPN is associated with transitions of the system net and the token nets. Based on the untimed version of the two-level colored Petri-net, the time concept of the two-level timed colored Petri-net can be defined as follows.

A *timed system net* is a tuple TSN = (SN (= (Σ, P, T, A, N, C, E, I)), W) = (Σ, P, T, A, N, C, E, I, W), where:

SN	CP-net;
$W: T \rightarrow \mathbb{R}$	time function defined on the set of transitions,

and a *timed token net* is a tuple $TTN_j = (TN_j (= (P_j, T_j, A_j, E_j, I_j)), W_j) = (P_j, T_j, A_j, E_j, I_j, W_j)$, where:

TN_j	PT-net;
$W_j: T_j \rightarrow \mathbb{R}$	time function defined on the set of transitions**.**

In the above two definitions, the firing time of the timed transitions $t \in T$ and $t_j \in T_j$ is assumed to be exponentially distributed with parameters W(t) and $W_j(t_j)$ respectively. Note that W(t) = 0 and $W_j(t_j) = 0$ if t and t_j are immediate transitions. In practice, we use different distribution functions during the simulation for modeling of different aspects of the stochastic job shops in order to make models more realistic.

D.2.2. Structure of the extended two-level CP-net

A *modular two-level CP net* is a tuple MOS = (MTSN, MTTN, SSP), where:

MTSN	modular timed system net;

$MTTN = \{MTTN_j \mid 1 \le j \le n\}$ set of modular timed token nets;

$SSP = \{SSP_i \mid 1 \le i \le m\}$ set of synchronized synchronization ports.

Let us consider the *modular timed system net* in more detail. A modular timed system net contains a finite set of modules, each of which is a timed CP-net with a module interface MI. These modules have disjoint sets of places, transitions and arcs. The module interfaces contain synchronization and communication ports that are shared places and transitions only. The *synchronization ports assignment function* SPA maps a synchronization port with corresponding private or shared transitions. As was mentioned before, there are three methods to organize the interaction of modules in the XCP-nets: place fusion sets, transition fusion sets and additional arcs. Each *place fusion set* is a set of places to be fused together. In the definitions below, 2^P denotes the set of all subsets of places. Each place fusion set can contain only shared places that are members of modules interfaces. It is necessary that all elements of a place fusion set have the same color set and have the same initial markings. In a similar manner, each *transition fusion set* is a set of transitions to be fused together and can contain only shared transitions that are members of module interfaces. It is necessary that all elements of a transition fusion set have the same time parameter W(t). Each *additional arc* connects modules by joining communication ports with different node types. Finally, the set of *synchronization ports* is the set of all synchronization ports of all modules. This set is used for subsequent synchronization transitions from the system net and token nets.

Now we give the formal definitions for the structure of the modular timed system net. The *modular timed system net* is a tuple MTSN = (S, PF, TF, AA, SP) satisfying

(i) S is a finite set of *modules* such that

– each module, $s \in S$, is a timed CP-net that can be represented as $M_s = (\Sigma_s, P_s, T_s, A_s, N_s, C_s, E_s, I_s, W_s, MI_s, SPA_s)$;

– the sets of net elements are pair wise disjoint: $s_1 \neq s_2 \Rightarrow (P_{s1} \cup T_{s1} \cup A_{s1}) \cap (P_{s2} \cup T_{s2} \cup A_{s2}) = \varnothing$ for all $s_1, s_2 \in S$;

– MI_s is a module interface: $MI_s = CP_s \cup SP_s$, where SP_s is a set of synchronization ports and CP_s is a set of communication ports of the module. Communication ports are shared places and transitions only: $CP_s = IP_s \cup IT_s$, and $P_s = PP_s \cup IP_s$ and $PP_s \cap IP_s = \varnothing$, $T_s = PT_s \cup IT_s$ and $PT_s \cap IT_s = \varnothing$, where PP_s/PT_s is a set of private places/transitions and IP_s/IT_s is a set of shared places/transitions;

– $SPA_s \in [SP_s \rightarrow T_s]$ is a *synchronization port assignment function* for module M_s that maps synchronization ports to transitions.

(ii) $PF \subseteq 2^P$ is a finite set of *place fusion sets*, where

– all members of a place fusion set are communication ports $PF \subset CP$;

– the members of a place fusion set have identical color sets and the same initialization expression; $C(p_1) = C(p_2) \wedge (p_1) = I(p_2)$ for all $p_1, p_2 \in pf$, where pf is a place fusion set. Here C(p) and I(p) are the *global* version (defined on the whole net) of a color function and initialization function respectively. The disjointness of the net elements implies that we can use a global color set function C and initialization function I instead of the local functions without ambiguity. The global functions are defined from the *local* functions (defined on a module) and can be represented as $C(p) = C_s(p)$ and $I(p) = I_s(p)$ for all $s \in S$ and $p \in P_s$.

(iii) $TF \subseteq 2^T$ is a finite set of *transition fusion sets*, such that

– all members of a transition fusion set are communication ports;

– the members of a transition fusion set have identical time parameter $W(t_1) = W(t_2)$ for all $t_1, t_2 \in tf$, where tf is a transition fusion set.

(iv) AA is a finite set of *additional arcs* such that $AA \cup A_s = \varnothing$ for all $s \in S$.

(v) SP is a finite set of *synchronization ports* of the system net such that $SP = \bigcup_{s \in S} SP_s$.

The notations used for the definition of modular timed token nets are similar, but somewhat simpler due to the absence of colors. The structure of the modular timed token nets is defined as follows. The *modular timed token net* is a tuple $MTTN_j = (S_j, PF_j, TF_j, AA_j, SP_j)$ satisfying

(i) S_j is a finite set of *modules* such that

– each module M_s, $s \in S_j$, is a timed PT-net $M_s = (P_{js}, T_{js}, A_{js}, E_{js}, I_{js}, W_{js}, MI_{js}, SPA_{js})$;

– the sets of net elements are pair wise disjoint: $s_1 \neq s_2 \Rightarrow (P_{js1} \cup T_{js1} \cup A_{js1}) \cap (P_{js2} \cup T_{js2} \cup A_{js2}) = \varnothing$ for all $s_1, s_2 \in S_j$;

– MI_{js} is a *module interface* defined in a similar way to the module interface for the system net;

– $SPA_{js} \in [SP_{js} \rightarrow T_{js}]$ is a *synchronization port assignment* function.

(ii) $PF_j \subseteq 2^P$ is a finite set of *place fusion sets* such that the members of a place fusion set have the same initialization expression; $I(p_1) = I(p_2)$ for all $p_1, p_2 \in pf$.

(iii) $TF_j \subseteq 2^T$ is a finite set of *transition fusion sets* such that the members of a transition fusion set have identical time parameter $W(t_1) = W(t_2)$ for all $t_1, t_2 \in tf \in TF$.

(iv) AA_j is a finite set of *additional arcs* such that $AA_j \cup A_{js} = \varnothing$ for all $s \in S_j$.

(v) SP_j is a finite set of the *synchronization ports* of the token net such that $SP_j = \bigcup_{s \in S_j} SP_s$ for all $s \in S_j$.

In addition, we define two functions describing the relationship between communication ports. First, NT is a *node type function*, which is defined from CP into {place, transition}. Second, PT is a *port type function*, which is defined from CP into {input, output}. As was mentioned before, interaction of modules could be done by combining communication ports into fusion sets or connecting them by additional arcs. Rules for modules composition determined by the *ports composition function* $CF(p_1, p_2)$ are as follows:

– For all $p_1, p_2 \in CP$: $CF(p_1, p_2) = \begin{cases} PF & \text{, if } PT(p_1) \neq PT(p_2) \wedge NT(p_1) = NT(p_2) = \text{place} \\ TF & \text{, elseif } PT(p_1) \neq PT(p_2) \wedge NT(p_1) = NT(p_2) = \text{transition} \\ AA & \text{, elseif } PT(p_1) \neq PT(p_2) \wedge NT(p_1) \neq NT(p_2) \\ \varnothing & \text{, otherwise} \end{cases}$

This implies that only ports with opposite port types could be connected. If the two ports to be composed have the same node types, these nodes will be merged. Otherwise, they can be connected by an additional arc. Run-time modules composition in arc expressions in a system net is done according to the same rules. Finally, the *set of synchronized synchronization ports* SSP is a modular version of *synchronized transition steps* from two-level CP-nets and can be defined as follows: the *synchronized synchronization ports* is a set $SSP_i = \{sp_i, sp_{i1}, \ldots, sp_{in}\}$,

where

- sp_i is a synchronization port belonging to a system net and the transition returned by the global synchronization port assignment function belongs to the set of the system nets transitions; $SPA(sp_i) \in T$;
- $\{sp_i, sp_{i1}, \ldots, sp_{in}\}$ is a sub-set of synchronization ports belonging to the token nets that are synchronized together and with the sp_i port; $SPA(sp) \in T_j$ for all $sp \in \{sp_i, sp_{i1}, \ldots, sp_{in}\}$;
- $SPA(sp_1) \cap SPA(sp_2) = \varnothing$ for all $sp_1, sp_2 \in \{sp_i, sp_{i1}, \ldots, sp_{in}\}$, $sp_1 \neq sp_2$; all ports synchronized by SSP_i belong to different nets.

D.2.3. Behavior of the extended two-level CP-net

Now we are ready to consider the behavior of XCP-nets. Our goal is to find modular versions of notations for *binding*, *synchronized binding*, *synchronized step* and enabled synchronized step.

A *place group* $pg \subseteq P$ is an equivalence class of the smallest equivalence relation containing all pairs $(p_1, p_2) \in P \times P$, where $\exists pf \in PF$: $p_1, p_2 \in pf$. Also, a *transition group* $tg \subseteq T$ consists of either a single non-fused transition t or all the members of a transition fusion set $tf \in TF$. The set of all place groups and transition groups are denoted by PG and TG, respectively. The place groups defined above consist of a portion of the set of places. The place can be a member of at most one place group while a transition can be a member of several transition groups. Hereafter, we use terms with a prime in order to denote place groups and transition groups; p' and t'. Then, the notions of token elements and local markings of the system net and token nets should be reformulated considering place groups.

A *token element* is a pair (p', c) where $p' \in PG$ and $c \in C(p')$. The set of all token elements is denoted by TE. A *marking* of system net is a multi-set over TE and a marking of token net is a multi-set over PG_j. In the definition above, a marking of the XCP-net is similar to the marking of two-level CP-nets with respect to the new definitions of local markings.

Now, we introduce the grouped version of binding. In the following definitions we use the semantics of a *set of variables associated with transition* Var(t), and *type of variable* denoted by Type(v). A *binding* of a transition group t' is a function b defined on the variables of the transition group, $Var(t') = \bigcup_{t \in t'} Var(t)$, such that $b(v) \in Type(v)$ for all $v \in Var(t')$. Then, the set of all bindings are denoted by B(t').

Next, we will introduce a new notion of the synchronized binding element. To simplify the definition of a synchronized binding element, we will use the term synchronized transition step instead of the synchronized transition ports. Then, we can easily see that any set of synchronized synchronization ports belonging to the SSP can be mapped to the set of synchronized transitions belonging to the SY with the synchronization port assignment function SPA. The *synchronized binding element* is a pair $((t'_{SN}, b), t'_{TN})$, where (t'_{SN}, b) is a binding element for the modular system net and t'_{TN} is a transition group from the token net, such that

(a) t'_{SN} and t'_{TN} belong to the same synchronized transition step SY_n.

(b) $t'_{TN} \subseteq T_j$ of the corresponding TN_j and $b(v_k) = TN_j$ for some $v_k \in Var(t'_{SN})$.

The set of all synchronized binding elements is denoted by BES. A *synchronized step* YS is a *non-empty* and *finite* multi-set over BES. A synchronized step YS is *enabled* in a marking M if the following properties are satisfied $(t'_{SN}, b) \in Y_{SN}$, $t'_{TN} \in Y_{TN}$ for $(t'_{SN}, b), t'_{TN}) \in YS$, where Y_{SN} and Y_{TN} are local steps in the system net and token net respectively. Here, we say that Y_{SN} and Y_{TN} are enabled in the marking M.

Finally, the definition of the synchronized step occurrence in the XCPN is identical to the definition of the synchronized step occurrence for two-level CP-nets (Sakara 2006).

Appendix E. IRSR model of the example QRS in chapter III

This section represents the complete information of the example QRS in chapter III, which is used for the simulations in this thesis.

E.1. Remanufacturing shop model

The three facility sets for workstations, resources, and buffers of $\boldsymbol{M^R}$ are as follows:

$\boldsymbol{W}$ = {[WS1], [WS2], [WS3], [WS4], [WS5], [WS6], [WS7], [WS8]};

$\boldsymbol{R}$ = {[RC1_1], [RC1_2], [RC2_1], [RC2_2], [RC2_3], [RC3_1], [RC3_2], [RC3_3], [RC4_1], [RC5_1], [RC5_2], [RC6_1], [RC6_2], [RC6_3], [RC7_1], [RC8_1]};

$\boldsymbol{B}$ = {[BF1], [BF2], [BF3]}.

The relationship functions among workstations, resources, and buffers of $\boldsymbol{M^R}$ are as follows:

$\lambda^{ra}(\boldsymbol{r}\in\boldsymbol{R})$ = [WSn] for $\boldsymbol{r}$ = [RCn_i];

$$\lambda^{ib}(\boldsymbol{w}\in\boldsymbol{W}) = \begin{pmatrix} [\text{BF1}] & \textbf{\textit{, if }}\ \boldsymbol{w}=[\text{WS1}] \\ [\text{BF2}] & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\lambda^{ob}(\boldsymbol{w}\in\boldsymbol{W}) = \begin{pmatrix} [\text{BF3}] & \textbf{\textit{, if }}\ \boldsymbol{w}=[\text{WS8}] \\ [\text{BF2}] & \textbf{\textit{, otherwise}} \end{pmatrix}.$$

The two value assignment functions for batch workstations and buffer capacities of $\boldsymbol{M^R}$ are as follows:

$$\nu^{bp}(\boldsymbol{w}\in\boldsymbol{W}) = \begin{pmatrix} \textbf{\textit{true}} & \textbf{\textit{, if }}\ \boldsymbol{w}=[\text{WS4}]\ \textbf{\textit{or}}\ [\text{WS5}] \\ \textbf{\textit{false}} & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\nu^{cb}(\boldsymbol{b}\in\boldsymbol{B}) = \begin{pmatrix} \textbf{\textit{infinite}} & \textbf{\textit{, if }}\ \boldsymbol{b}=[\text{BF1}] \\ 1{,}000 & \textbf{\textit{, elseif }}\ \boldsymbol{b}=[\text{BF2}] \\ 300 & \textbf{\textit{, otherwise}} \end{pmatrix},$$

and last, the three functions for specifying facility types of $\boldsymbol{M^R}$ are as follows:

$$\tau^{w}(\boldsymbol{w}\in\boldsymbol{W}) = \begin{pmatrix} \textbf{\textit{disassembly}} & \textbf{\textit{, if }}\ \boldsymbol{w}=[\text{WS1}],[\text{WS2}],\ \textbf{\textit{or}}\ [\text{WS3}] \\ \textbf{\textit{machinging}} & \textbf{\textit{, elseif }}\ \boldsymbol{w}=[\text{WS4}] \\ \textbf{\textit{cleaning}} & \textbf{\textit{, elseif }}\ \boldsymbol{w}=[\text{WS5}]\ \textbf{\textit{or}}\ [\text{WS6}] \\ \textbf{\textit{reassembly}} & \textbf{\textit{, elseif }}\ \boldsymbol{w}=[\text{WS7}] \\ \textbf{\textit{testing}} & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\tau^{br}(\boldsymbol{w}\in\boldsymbol{W}) = \begin{pmatrix} \textbf{\textit{human}} & \textbf{\textit{, if }}\ \boldsymbol{w}=[\text{WS2}],[\text{WS3}],[\text{WS7}],\ \textbf{\textit{or}}\ [\text{WS8}] \\ \textbf{\textit{machine}} & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\tau^{b}(b \in B) = \begin{pmatrix} EtS & ,if\ b = [\mathrm{BF1}] \\ Internal & ,elseif\ b = [\mathrm{BF2}] \\ EfS & ,otherwise \end{pmatrix}.$$

E.2. Product-alternative process model

The PDSP sets of M^{P} are as follows:

$P = P_{[A]} \cup P_{[B]}$;
$P^{U} = P^{U}_{[A]} \cup P^{U}_{[B]}$;
$P^{R} = P^{R}_{[A]} \cup P^{R}_{[B]}$,

and PDSP relationship functions are as follows:

$$\lambda^{d}(o \in \cup O_p) = \begin{pmatrix} \lambda^{d}_{[A]} & ,if\ p \text{ is a PDSP of [A]} \\ \lambda^{d}_{[B]} & ,otherwise \end{pmatrix};$$

$\lambda^{cq}(q \in \cup Q_p) = \cup \lambda^{cq}_{p}(q \in Q_p)$,

where

λ^{d}_{up} sub function of λ^{d} for the used product up;
λ^{cq}_{p} sub function of λ^{cq} for the corresponding PDSP p.

The consecutive subsections explain sub-elements for set unions and sub-functions for function application cases. But the v^{nor} modeling is omitted because of its complexity (see section VIII.3.3.3 for the detailed explanation.).

E.2.1. Product-alternative process model for [A]

The PDSP sets of the used product [A] is as follows:

$P_{[A]}$ = {[A], [A12], [A23], [A1], [A2], [A3], [A_R]};
$P^{U}_{[A]}$ = {[A]};
$P^{R}_{[A]}$ = {[A_R]}.

The possible qualities of the PDSPs of [A] are as follows:

$Q_{[A]}$ = {*ND, J12AH, A1CoW, A1CoW_A2CoG, A1CoW_J2CB, A1RST, A1RST_J2CB, J2CB, J2AH, A2SoG, J1AH, A3CRK_J1AH*};

$\boldsymbol{Q}_{[A12]}$ = {*ND*, *A1CoW*, *A1CoW_A2SoG*, *A1RST*, *A1RST_A2SoG*, *A2SoG*, *J1AH*};

$\boldsymbol{Q}_{[A23]}$ = {*ND*, *A2CoG*, *A2SoG*, *A2SoG_A3CRK*, *A3CRK*, *J2AH*, *J2CB*};

$\boldsymbol{Q}_{[A1]}$ = {*ND*, *CoW*, *RST*, *CoW*};

$\boldsymbol{Q}_{[A2]}$ = {*ND*, *CoG*, *SoG*, *UIQ*};

$\boldsymbol{Q}_{[A3]}$ = {*ND*, *CRK*};

$\boldsymbol{Q}_{[A\_R]}$ = {*ND*},

where

AH joint adherence;
CB crack or break of joint;
CoG crack or strong wear of groove;
CoW crack or wear;
CRK crack;
ND no defect;
RST rust;
SoG surface scratch or weak wear of groove;
UIQ unidentified quality.

The *A1*, *A2*, and *A3* placed right before the quality means the corresponding part shows the indicated defects, and the quality concatenated with '_' represents the PDSP has both defects; for example, the quality *A1CoW_J2CB* of the used product [A] means that it has the two defects: [A1] is cracked or worn and [AJ2] is cracked or broken. This naming rule is also applied to the used product [B]'s quality modeling. The quality relationships among PDSPs of [A] are as follows:

$$\boldsymbol{\lambda}^{cq}_{[A]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} A2SoG & \textbf{,if } \boldsymbol{p} = [\mathrm{A12}] \textbf{ and } \boldsymbol{q} = A2SoG \\ J1AH & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A12}] \textbf{ and } \boldsymbol{q} = (J1AH \textbf{ or } A3CRK\_J1AH) \\ A2CoG & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A23}] \textbf{ and } \boldsymbol{q} = A1CoW\_A2CoG \\ J2AH & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A23}] \textbf{ and } \boldsymbol{q} = J2AH \\ J2CB & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A23}] \textbf{ and } \boldsymbol{q} = (A1CoW\_J2CB, A1RST\_J2CB, \textbf{or } J2CB) \\ CoW & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A1}] \textbf{ and } \boldsymbol{q} = (A1CoW, A1CoW\_A2CoG, \textbf{or } A1CoW\_J2CB) \\ RST & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A1}] \textbf{ and } \boldsymbol{q} = (A1RST \textbf{ or } A1RST\_J2CB) \\ CRK & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A3}] \textbf{ and } \boldsymbol{q} = A3CRK\_J1AH \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\boldsymbol{\lambda}^{cq}_{[A12]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} CoW & \textbf{,if } \boldsymbol{p} = [\mathrm{A1}] \textbf{ and } \boldsymbol{q} = (A1CoW \textbf{ or } A1CoW\_A2SoG) \\ RST & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A1}] \textbf{ and } \boldsymbol{q} = (A1RST \textbf{ or } A1RST\_A2SoG) \\ SoG & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A2}] \textbf{ and } \boldsymbol{q} = (A1CoW\_A2SoG, A1RST\_A2SoG, \textbf{or } A2SoG) \\ UIQ & \textbf{,elseif } \boldsymbol{p} = [\mathrm{A2}] \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[A23]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} CoG & \textbf{,if } \boldsymbol{p} = [\text{A2}] \textbf{ and } \boldsymbol{q} = A2CoG \\ SoG & \textbf{,elseif } \boldsymbol{p} = [\text{A2}] \textbf{ and } \boldsymbol{q} = (A2SoG \textbf{ or } A2SoG\_A3CRK) \\ UIQ & \textbf{,elseif } \boldsymbol{p} = [\text{A2}] \\ CRK & \textbf{,elseif } \boldsymbol{p} = [\text{A3}] \textbf{ and } \boldsymbol{q} = (A3CRK \textbf{ or } A2SoG\_A3CRK) \\ ND & \textbf{,otherwise} \end{pmatrix}.$$

The operations to be processed of each PDSP of [A] are as follows:

$\boldsymbol{O}_{[A]}$ = {OPav1, OPa01, OPa02}, $\boldsymbol{o}^{s}_{[A]}$ = {OPav1}, $\boldsymbol{o}^{e}_{[A]}$ = {OPa01, OPa02};
$\boldsymbol{O}_{[A12]}$ = {OPa03}, $\boldsymbol{o}^{s}_{[A12]}$ = {OPa03}, $\boldsymbol{o}^{e}_{[A12]}$ = {OPa03};
$\boldsymbol{O}_{[A23]}$ = {OPa04}, $\boldsymbol{o}^{s}_{[A23]}$ = {OPa04}, $\boldsymbol{o}^{e}_{[A23]}$ = {OPa04};
$\boldsymbol{O}_{[A1]}$ = {OPav2, OPa05, OPa08}, $\boldsymbol{o}^{s}_{[A1]}$ = {OPav2}, $\boldsymbol{o}^{e}_{[A1]}$ = {OPa08};
$\boldsymbol{O}_{[A2]}$ = {OPa06, OPa07, OPa08}, $\boldsymbol{o}^{s}_{[A2]}$ = {OPa06}, $\boldsymbol{o}^{e}_{[A1]}$ = {OPa08};
$\boldsymbol{O}_{[A3]}$ = {OPa08}, $\boldsymbol{o}^{s}_{[A3]}$ = {OPa08}, $\boldsymbol{o}^{e}_{[A3]}$ = {OPa08};
$\boldsymbol{O}_{[A\_R]}$ = {OPa09}, $\boldsymbol{o}^{s}_{[A\_R]}$ = {OPa09}, $\boldsymbol{o}^{e}_{[A\_R]}$ = {OPa09},

and the relationships among them and PDSPs of [A] are as follows:

$$\lambda^{no}_{[A*]}(\boldsymbol{o} \in \boldsymbol{O}_{[A*]}) = \begin{pmatrix} \{\text{OPa01, OPa02}\} & \textbf{,if } \boldsymbol{o} = \text{OPav1} \\ \{\text{OPa05, OPa08}\} & \textbf{,elseif } \boldsymbol{o} = \text{OPav2} \\ \{\text{OPa07}\} & \textbf{,elseif } \boldsymbol{o} = \text{OPa06} \\ \{\text{OPa08}\} & \textbf{,elseif } \boldsymbol{o} = (\text{OPa05} \textbf{ or } \text{OPa07}) \\ \textbf{null} & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{d}_{[A]}(\boldsymbol{o} \in \cup \boldsymbol{O}_p) = \begin{pmatrix} \{[\text{A1}], [\text{A23}]\} & \textbf{,if } \boldsymbol{o} = \text{OPa01} \\ \{[\text{A12}], [\text{A3}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPa02} \\ \{[\text{A1}], [\text{A2}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPa03} \\ \{[\text{A2}], [\text{A3}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPa04} \\ \{[\text{A\_R}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPa08} \\ \textbf{null} & \textbf{,otherwise} \end{pmatrix}.$$

The operation processing statistical information including their correlation of [A] is as follows:

$\Delta^{op}_{[A]}$ = {(OPa01, *ND*, (Normal(9.6, 1), 0, 0, .062), ∅, {(*A1CoW*, 1)}, {(*ND*, .8), (*A1RST*, .2)}), (OPa01, *J12AH*, (Normal(15.4, 1.5), .042, 0, .958), ∅, {(*A1CoW_A2CoG*, 1)}, ∅), (OPa01, *J2CB*, (Normal(9.6, 1), 0, 0, .054), ∅, {(*A1CoW_J2CB*, 1)}, {(*J2CB*, .8), (*A1RST_J2CB*, .2)}), (OPa02, *ND*, (Normal (9.3, .9), 0, 0, .098), ∅, {(*A2SoG*, 1)}, ∅), (OPa02, *J12AH*, (Normal(16.7, 1.7), .032, 0, .027), ∅, {(*A3CRK_J1AH*, 1)}, ∅), (OPa02, *J2CB*, (Normal(4.2, .4), 0, 0, 0), ∅, ∅, {(*ND*, 1)})};
$\Delta^{op}_{[A12]}$ = {(OPa03, *ND*, (Normal(6.4, .6), .021, .017, .048), {(*A2SoG*, 1)}, {(*A1CoW*, 1)}, {(*ND*, .8), (*A1RST*, .2)}), (OPa03, *A2SoG*, (Normal(6.4, .6), 0, 0, .254), ∅, {(*A1CoW_A2SoG*, .3), (*A1RST_A2SoG*, .7)}, ∅), (OPa03, *J1AH*, (Normal(1.2, 1), 0, 0, 1), ∅, {(*A1CoW_A2SoG*, 1)}, ∅)};

$\Delta^{op}_{[A23]}$ = {(OPa04, *ND*, (Normal(8.9, .9), .042, 0, .068), ∅, {(*A2SoG*, .5), (*A3CRK*, .5)}, ∅), (OPa04, *A2CoG*, (Normal(8.9, .9), 0, 0, .128), ∅, {(*A2SoG_A3CRK*, 1)}, ∅), (OPa04, *J2AH*, (Normal(16, 1.6), .032, 0, .027), ∅, {(*A3CRK*, .4), (*A2SoG_A3CRK*, .6)}, {(*ND*, 1)}), (OPa04, *J2CB*, (Normal(6.1, .6), 0, 0, 0), ∅, ∅, {(*ND*, 1)})};

$\Delta^{op}_{[A1]}$ = {(OPa05, *RST*, (Deterministic(6.4), .050, 0, 0), ∅, ∅, {(*ND*, 1)}), (OPa08, *ND*, (Normal(8.7, .9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A2]}$ = {(OPa06, *UIQ*, (Deterministic(16.2), .032, 0, 0), ∅, ∅, {(*ND*, .8), (*SoG*, .2)}), (OPa06, *SoG*, (Deterministic(16.2), .071, 0, 0), ∅, ∅, ∅), (OPa07, *SoG*, (Normal(4.4, .4), .053, 0, .104), ∅, {(*CoG*, 1)}, {(*ND*, 1)}), (OPa08, *ND*, (Normal(8.7, .9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A3]}$ = {(OPa08, *ND*, (Normal(8.7, .9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A\_R]}$ = {(OPa09, *ND*, (Deterministic(2.8), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{cr}_{[A]}$ = {(OPa01, {(*, *, ***false***, 1.3), (*ND*, *A1CoW*, ***true***, .7), (*J12AH*, *A1CoW*, ***true***, .7)})};

$\Delta^{cr}_{[A2]}$ = {(OPa07, {(*SoG*, *CoG* , *, .8)})}.

The functions for specifying the necessity of quality test and the limit of rework times of [A] are as follows:

$$\cup v^{qt}_{[A*]}(o \in O_{[A*]}) = \begin{pmatrix} \textbf{\textit{false}} & \textbf{\textit{, if }}\ o = \text{OPa09} \\ \textbf{\textit{true}} & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\cup v^{rw}_{[A*]}(o \in O_{[A*]}) = \begin{pmatrix} \textbf{0} & \textbf{\textit{, if }}\ o = \text{OPa08}\ \textbf{\textit{or}}\ \text{OPa09} \\ \textbf{1} & \textbf{\textit{, elseif }}\ o = \text{OPa07} \\ \textbf{2} & \textbf{\textit{, elseif }}\ o = \text{OPa05}\ \textbf{\textit{or}}\ \text{OPa06} \\ \textbf{\textit{inf.}} & \textbf{\textit{, otherwise}} \end{pmatrix},$$

and the operation type specification function is as follows:

$$\tau^{o}_{[A*]}(o \in O_{[A*]}) = \begin{pmatrix} \textbf{\textit{virtual}} & \textbf{\textit{, if }}\ o = \text{OPav1}\ \textbf{\textit{or}}\ \text{OPav2} \\ \textbf{\textit{disassembly}} & \textbf{\textit{, elseif }}\ o = \text{OPa01, OPa02, OPa03,}\ \textbf{\textit{or}}\ \text{OPa04} \\ \textbf{\textit{cleaning}} & \textbf{\textit{, elseif }}\ o = \text{OPa05}\ \textbf{\textit{or}}\ \text{OPa06} \\ \textbf{\textit{machinging}} & \textbf{\textit{, elseif }}\ o = \text{OPa07} \\ \textbf{\textit{reassembly}} & \textbf{\textit{, elseif }}\ o = \text{OPa08} \\ \textbf{\textit{testing}} & \textbf{\textit{, otherwise}} \end{pmatrix}.$$

Last, the arrival information of [A] is as follows:

$\Delta^{pa}_{[A]}$ = (Poisson(500), Normal(25, 2.5), {(*ND*, .7), (*J12AH*, .2), (*J2CB*, .1)}).

E.2.2. Product-alternative process model for [B]

The PDSP sets of the used product [B] is as follows:

$\boldsymbol{P}_{[B]}$ = {[B], [B2345] , [B234] , [B345] , [B34] , [B1] , [B2] , [B3] , [B4] , [B5] , [B_R]};

$\boldsymbol{P}^{U}_{[B]}$ = {[B]};
$\boldsymbol{P}^{R}_{[B]}$ = {[B_R]}.

The possible qualities of PDSPs of [B] are as follows:

$\boldsymbol{Q}_{[B]}$ = {*ND, B1CRK, B1CRK_J3PA, B1CRK_J5BB, B1RST_J3PA, J5BB, J1BA, B1CRK_J1BA*};
$\boldsymbol{Q}_{[B2345]}$ = {*ND, B2CRK, B2CRK_B4DFM, B4DFM, B4DFM_J5BB, B5CoTA, B5CoTA_J3BA, B5CoTA_J3BA_J4TF, B5CoTA_J3PA, B5CoTA_J4TF, J3BA, J3PA, J4TF, J4TF_J5BB, J5BB, B5RST, B5RST_J3PA, B5RST_J3PA*};
$\boldsymbol{Q}_{[B234]}$ = {*ND, B2CRK, B2CRK_B4DFM, B3CoWoD, B3CoWoD_J4TF, B4DFM, B4DFM_J4TF, J3BA, J3BA_J4TF, J3PA, J4TF, J5BB, B4DFM_J5BB, J4TF_J5BB*};
$\boldsymbol{Q}_{[B345]}$ = {*ND, B4DFM, B4DFM_B5CoTA, B4DFM_J5BB, B5CoTA, B5CoTA_J4TF, J4TF, J4TF_J5BB, J5BB, B5RST, J3BA, J3BA_J5BB*};
$\boldsymbol{Q}_{[B34]}$ = {*ND, B3CoWoD, B3CoWoD_B4RST, B3CoWoD_J4TF, B4DFM, B4DFM_J4TF, B4RST, J4TF*};
$\boldsymbol{Q}_{[B1]}$ = {*ND, CRK, RST*};
$\boldsymbol{Q}_{[B2]}$ = {*ND, CoHW, UIQ*};
$\boldsymbol{Q}_{[B3]}$ = {*ND, CoWoD*};
$\boldsymbol{Q}_{[B4]}$ = {*ND, DFM, RST*};
$\boldsymbol{Q}_{[B5]}$ = {*ND, CoTA, RST*};
$\boldsymbol{Q}_{[B\_R]}$ = {*ND*}.

where

BA	bolt abrasion;
BB	bolt break;
CoHW	crack or wear of the hole;
CoTA	crack or teeth abrasion;
CoWoD	crack, wear of both ends, or deformation;
DFM	deformation;
PA	part adherence;
TF	tight fitting.

The unexplained abbreviations have the same meaning with that in the qualities of the used product [A]. The quality relationships among PDSPs of [A] are as follows:

$$\boldsymbol{\lambda}^{cq}_{[B]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} J3PA & \textbf{, if } \boldsymbol{p} = [\mathrm{B2345}] \textbf{ and } \boldsymbol{q} = (B1CRK\_J3PA \textbf{ or } B1RST\_J3PA) \\ J5BB & \textbf{, elseif } \boldsymbol{p} = [\mathrm{B2345}] \textbf{ and } \boldsymbol{q} = (B1CRK\_J5BB, \textbf{ or } J5BB) \\ CRK & \textbf{, elseif } \boldsymbol{p} = [\mathrm{B1}] \textbf{ and } \boldsymbol{q} = (B1CRK, B1CRK\_J3PA, \textbf{ or } B1CRK\_J5BB) \\ RST & \textbf{, elseif } \boldsymbol{p} = [\mathrm{B1}] \textbf{ and } \boldsymbol{q} = B1RST\_J3PA \\ ND & \textbf{, otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[B2345]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} J3BA & \textbf{,if } \boldsymbol{p} = [\text{B234}] \textbf{ and } \boldsymbol{q} = (B5CoTA\_J3BA, B5RST\_J3BA, \textbf{or } J3BA) \\ J3BA\_J4TF & \textbf{,elseif } \boldsymbol{p} = [\text{B234}] \textbf{ and } \boldsymbol{q} = B5CoTA\_J3BA\_J4TF \\ J3PA & \textbf{,elseif } \boldsymbol{p} = [\text{B234}] \textbf{ and } \boldsymbol{q} = (B5CoTA\_J3PA, B5RST\_J3PA, \textbf{or } J3PA) \\ J4TF & \textbf{,elseif } \boldsymbol{p} = [\text{B234}] \textbf{ and } \boldsymbol{q} = B5CoTA\_J4TF \\ B4DFM & \textbf{,elseif } \boldsymbol{p} = [\text{B345}] \textbf{ and } \boldsymbol{q} = (B2CRK\_B4DFM \textbf{ or } B4DFM) \\ \boldsymbol{q} & \textbf{,elseif } \boldsymbol{p} = [\text{B345}] \textbf{ and } \boldsymbol{q} = (B4DFM\_J5BB, J4TF, J4TF\_J5BB, \textbf{or } J5BB) \\ CoHW & \textbf{,elseif } \boldsymbol{p} = [\text{B2}] \textbf{ and } \boldsymbol{q} = (B2CRK \textbf{ or } B2CRK\_B4DFM) \\ UIQ & \textbf{,elseif } \boldsymbol{p} = [\text{B2}] \\ RST & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} = (B5RST, B5RST\_J3PA, \textbf{or } B5RST\_J3BA) \\ CoTA & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} \neq (J3BA, J3PA \textbf{ or } ND) \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[B234]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} B4DFM & \textbf{,if } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} = B2CRK\_B4DFM \\ \boldsymbol{q} & \textbf{,elseif } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} \neq B2CRK \\ CoHW & \textbf{,elseif } \boldsymbol{p} = [\text{B2}] \textbf{ and } \boldsymbol{q} = (B2CRK \textbf{ or } B2CRK\_B4DFM) \\ UIQ & \textbf{,elseif } \boldsymbol{p} = [\text{B2}] \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[B345]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} B4DFM & \textbf{,if } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} = (B4DFM \textbf{ or } B4DFM\_B5CoTA) \\ J4TF & \textbf{,elseif } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} = (J4TF \textbf{ or } B5CoTA\_J4TF) \\ CoTA & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} = (B4DFM\_B5CoTA, B5CoTA, \textbf{or } B5CoTA\_J4TF) \\ RST & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} = B5RST \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[B345]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} B4DFM & \textbf{,if } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} = (B4DFM \textbf{ or } B4DFM\_B5CoTA) \\ J4TF & \textbf{,elseif } \boldsymbol{p} = [\text{B34}] \textbf{ and } \boldsymbol{q} = (J4TF \textbf{ or } B5CoTA\_J4TF) \\ CoTA & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} = (B4DFM\_B5CoTA, B5CoTA, \textbf{or } B5CoTA\_J4TF) \\ RST & \textbf{,elseif } \boldsymbol{p} = [\text{B5}] \textbf{ and } \boldsymbol{q} = B5RST \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[B34]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} CWD & \textbf{,if } \boldsymbol{p} = [\text{B3}] \textbf{ and } \boldsymbol{q} = (B3CWD \textbf{ or } B3CWD\_B4RST) \\ RST & \textbf{,elseif } \boldsymbol{p} = [\text{B4}] \textbf{ and } \boldsymbol{q} = (B3CWD\_B4RST \textbf{ or } B4RST) \\ DFM & \textbf{,elseif } \boldsymbol{p} = [\text{B4}] \textbf{ and } \boldsymbol{q} = B4DFM \\ ND & \textbf{,otherwise} \end{pmatrix}.$$

The operations to be processed of each PDSP of [B] are as follows:

$\boldsymbol{O}_{[B]}$ = {OPb01},	$\boldsymbol{o}^{s}_{[B]}$ = {OPb01},	$\boldsymbol{o}^{e}_{[B]}$ = {OPb01};
$\boldsymbol{O}_{[B2345]}$ = {OPbv1, OPb02, OPb03},	$\boldsymbol{o}^{s}_{[B2345]}$ = {OPbv1},	$\boldsymbol{o}^{e}_{[B2345]}$ = {OPb02, OPb03};
$\boldsymbol{O}_{[B234]}$ = {OPb04},	$\boldsymbol{o}^{s}_{[B234]}$ = {OPb04},	$\boldsymbol{o}^{e}_{[B234]}$ = {OPb04};
$\boldsymbol{O}_{[B345]}$ = {OPb05},	$\boldsymbol{o}^{s}_{[B345]}$ = {OPb05},	$\boldsymbol{o}^{e}_{[B345]}$ = {OPb05};
$\boldsymbol{O}_{[B34]}$ = {OPb06},	$\boldsymbol{o}^{s}_{[B34]}$ = {OPb06},	$\boldsymbol{o}^{e}_{[B34]}$ = {OPb06};
$\boldsymbol{O}_{[B1]}$ = {OPbv2, OPb07, OPb14},	$\boldsymbol{o}^{s}_{[B1]}$ = {OPbv2},	$\boldsymbol{o}^{e}_{[B1]}$ = {OPb07, OPb14};
$\boldsymbol{O}_{[B2]}$ = {OPb08, OPb09, OPb14},	$\boldsymbol{o}^{s}_{[B2]}$ = {OPb08},	$\boldsymbol{o}^{e}_{[B2]}$ = {OPb14};
$\boldsymbol{O}_{[B3]}$ = {OPb10, OPb11, OPb14},	$\boldsymbol{o}^{s}_{[B3]}$ = {OPb10},	$\boldsymbol{o}^{e}_{[B3]}$ = {OPb14};
$\boldsymbol{O}_{[B4]}$ = {OPb12, OPb14},	$\boldsymbol{o}^{s}_{[B4]}$ = {OPb12},	$\boldsymbol{o}^{e}_{[B4]}$ = {OPb14};
$\boldsymbol{O}_{[B5]}$ = {OPb13, OPb14},	$\boldsymbol{o}^{s}_{[B5]}$ = {OPb13},	$\boldsymbol{o}^{e}_{[B5]}$ = {OPb14};

$\boldsymbol{O}_{[B\_R]}$ = {OPb15}, $\boldsymbol{o}^{s}_{[B\_R]}$ = {OPb15}, $\boldsymbol{o}^{e}_{[B\_R]}$ = {OPb15},

and the relationships among them and PDSPs of [B] are as follows:

$$\boldsymbol{\lambda}^{no}_{[B*]}(\boldsymbol{o}\in\boldsymbol{O}_{[B*]}) = \begin{pmatrix} \{\text{OPb02, OPb03}\} & \textbf{,if } \boldsymbol{o} = \text{OPbv1} \\ \{\text{OPb07, OPb14}\} & \textbf{,elseif } \boldsymbol{o} = \text{OPbv2} \\ \{\text{OPb09}\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb08} \\ \{\text{OPb11}\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb10} \\ \{\text{OPb14}\} & \textbf{,elseif } \boldsymbol{o} = \textbf{(}\text{OPb07, OPb09, OPb11, OPb12, }\textbf{or}\text{ OPb13}\textbf{)} \\ \textbf{null} & \textbf{,otherwise} \end{pmatrix};$$

$$\boldsymbol{\lambda}^{d}_{[B]}(\boldsymbol{o}\in\cup\boldsymbol{O}_{p}) = \begin{pmatrix} \{[\text{B1}],[\text{A2345}]\} & \textbf{,if } \boldsymbol{o} = \text{OPb01} \\ \{[\text{A345}],[\text{A2}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb02} \\ \{[\text{A234}],[\text{A5}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb03} \\ \{[\text{A34}],[\text{A2}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb04} \\ \{[\text{A34}],[\text{A5}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb05} \\ \{[\text{A3}],[\text{A4}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb06} \\ \{[\text{A\_R}]\} & \textbf{,elseif } \boldsymbol{o} = \text{OPb14} \\ \textbf{null} & \textbf{,otherwise} \end{pmatrix}.$$

The operation processing statistical information including their correlation of [B] is as follows:

$\boldsymbol{\Delta}^{op}_{[B]}$ = {(OPb01, *ND*, (Normal(2.3, .2), 0, 0, .062), ∅, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1), (*B1CRK_J5BB*, .1)}, {(*ND*, .8), (*B1RST_J3PA*, .1), (*J5BB*, .1)}), (OPb01, *J1BA*, (Normal(5.1, .5), .084, .032, .175), {(*B1CRK_J1BA*, 1)}, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1), (*B1CRK_J5BB*, .1)}, {(*ND*, .8), (*B1RST_J3PA*, .1), (*J5BB*, .1)}), (OPb01, *B1CRK_J1BA*, (Normal(5.1, .5), .062, 0, 0), ∅, ∅, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1),(*B1CRK_J5BB*, .1)}), (OPb01, *ND*, (Normal(2.3, .2), 0, 0, .062), ∅, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1), (*B1CRK_J5BB*, .1)}, {(*ND*, .8), (*B1RST_J3PA*, .1), (*J5BB*, .1)}), (OPb01, *J1BA*, (Normal(5.1, .5), .084, .032, .175), {(*B1CRK_J1BA*, 1)}, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1), (*B1CRK_J5BB*, .1)}, {(*ND*, .8), (*B1RST_J3PA*, .1), (*J5BB*, .1)}), (OPb01, *B1CRK_J1BA*, (Normal(5.1, .5), .062, 0, 0), ∅, ∅, {(*B1CRK*, .8), (*B1CRK_J3PA*, .1),(*B1CRK_J5BB*, .1)})};

$\boldsymbol{\Delta}^{op}_{[B2345]}$ = {(OPb02, *ND*, (Normal(12.6, 1.3), 0, .271, 0), {(*J3BA*, 1)}, ∅, {(*ND*, .8), (*J4TF*, .2)}), (OPb02, *J3PA*, (Normal(2.2, 2), 0, 0, .352), ∅, {(*B4DFM*, .3), (*B2CRK*, .5), (*B2CRK_B4DFM*, .2)},{(*ND*, .6), (*J4TF*, .4)}), (OPb02, *J5BB*, (Normal(12.6, 1.3), 0, .271, 0), {(*J3BA_J5BB*, 1)}, ∅, {(*J5BB*, .8), (*J4TF_J5BB*, .2)}), (OPb02, *J3BA*, (Normal(16.4, 1.6), 0, 0, .406), ∅, {(*B4DFM*, 1)}, {(*ND*, .8), (*J4TF*, .2)}), (OPb02, *J3BA_J5BB*, (Normal(16.4, 1.6), 0, 0, .406), ∅, {(*B4DFM_J5BB*, 1)}, {(*J5BB*, .8),(*J4TF_J5BB*, .2)}), (OPb03, *ND*, (Normal(6.1, .6), .103, 0, 0), ∅, ∅, {(*ND*, .8), (*B5CoTA*, .1), (*B5RST*, .1)}), (OPb03, *J3PA*, (Normal(6.1, .6), .103, 0, 0), ∅, ∅, {(*J3PA*, .8), (*B5CoTA_J3PA*, .1), (*B5RST_J3PA*, .1)}), (OPb03, *J3BA*, (Normal(6.1, .6), .103, 0, 0), ∅, ∅, {(*J3BA*, .8), (*B5CoTA_J3BA*, .1), (*B5RST_J3BA*, .1)}), (OPb03, *J5BB*, (Normal(6.1, .6), 0, 0, 1), ∅, {(*B5CoTA*, .8),

(*B5CoTA_J4TF*, .2)}, ∅), (OPb03, *J3BA_J5BB*, (Normal(11, 1.1), 0, 0, 1), ∅, {(*B5CoTA_J3BA*, .8), (*B5CoTA_J3BA_J4TF*, .2)}, ∅)};

$\Delta^{op}_{[B234]}$ = {(OPb04, *ND*, (Normal(10.3, 1.0), 0, .214, .084), {(*J3BA*, 1)}, {(*B4DFM*, 1)}, {(*ND*, .9), (*B3CWD*, .1)}), (OPb04, *J3BA*, (Normal(16.5, 1.7), 0, 0, .447), ∅, {(*B4DFM*, 1)}, {(*ND*, .9), (*B3CWD*, .1)}), (OPb04, *J3BA_J4TF*, (Normal(16.5, 1.7), 0, 0, .447), ∅, {(*B4DFM_J4TF*, 1)}, {(*ND*, .9), (*B3CWD*, .1)}), (OPb04, *J3PA*, (Normal(13.4, 1.3), 0, 0, .387), ∅, {(*B4DFM*, .3), (*B2CRK*, .5), (*B2CRK_B4DFM*, .2)}, {(*ND*, .9), (*B3CWD*, .1)}), (OPb04, *J4TF*, (Normal(10.3, 1.0), 0, .214, .084), {(*J3BA*, 1)}, {(*B4DFM_J4TF*, 1)}, {(*J4TF*, .9), (*B3CWD_J4TF*, .1)})};

$\Delta^{op}_{[B345]}$ = {(OPb05, *ND*, (Normal(6.9, .7), .037, .026, .059), {(*J5BB*, 1)}, {(*B5CoTA*, 1)}, {(*ND*, .8), (*B5RST*, .2)}), (OPb05, *B4DFM*, (Normal(6.9, .7), .037, .026, .059), {(*B4DFM_J5BB*, 1)}, {(*B4DFM_B5CoTA*, 1)}, {(*ND*, .8), (*B5RST*, .2)}), (OPb05, *B4DFM_J5BB*, (Normal(12.4, 1.2), 0, 0, 1), ∅, {(*B4DFM_B5CoTA*, 1)}, {(*B4DFM*, 1)}), (OPb05, *J4TF*, (Normal(6.9, .7), 0, .020, .038), {(*J4TF_J5BB*, 1)}, {(*B5CoTA_J4TF*, 1)}, ∅), (OPb05, *J4TF_J5BB*, (Normal(12.4, 1.2), 0, 0, 1), ∅, {(*B5CoTA_J4TF*, 1)}, {(*J4TF*, 1)}), (OPb05, *J5BB*, (Normal(12.4, 1.2), 0, 0, 1), ∅, {(*B5CoTA*, 1)}, {(*ND*, 1)})};

$\Delta^{op}_{[B34]}$ = {(OPb06, *ND*, (Normal(1.9, .2), 0, 0, .034), ∅, {(*B3CWD*, 1)}, {(*ND*, .8), (*B4RST*, .2)}), (OPb06, *B3CWD*, (Normal(1.9, .2), 0, 0, 0), ∅, ∅, {(*B3CWD*, .8), (*B3CWD_B4RST*, .2)}), (OPb06, *B3CWD_J4TF*, (Normal(2.4, .2), .510, 0, 0), ∅, ∅, {(*B3CWD*, .8), (*B3CWD_B4RST*, .2)}), (OPb06, *B4DFM*, (Normal(1.9, .2), 0, 0, 0), ∅, ∅, ∅), (OPb06, *B4DFM_J4TF*, (Normal(2.4, .2), .510, 0, 0), ∅, ∅, {(*B4DFM*, 1)}), (OPb06, *J4TF*, (Normal(2.4, .2), .510, 0, 0), ∅, ∅, {(*ND*, .8), (*B4RST*, .2)})};

$\Delta^{op}_{[B1]}$ = {(OPb07, *RST*, (Deterministic(3.4), .105, 0, 0), ∅, ∅, {(*ND*, 1)}), (OPb14, *ND*, (Normal(11.5, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B2]}$ = {(OPb08, *UIQ*, (Deterministic(20.8), .183, 0, .014), ∅, {(*CoHW*, 1)}, {(*ND*, 1)}), (OPb09, *ND*, (Normal(9.7, 1), 0, 0, .130), ∅, {(*CoHW*, 1)}, ∅), (OPb14, *ND*, (Normal(11.5, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B3]}$ = {(OPb10, *ND*, (Normal(8.6, .9), 0, 0, .105), ∅, {(*CWD*, 1)}, ∅), (OPb11, *ND*, (Deterministic(18.9), .1, 0, 0), ∅, ∅, ∅), (OPb14, *ND*, (Normal(11.5, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B4]}$ = {(OPb12, *ND*, (Deterministic(13.7), 0, 0, 0), ∅, ∅, ∅), (OPb12, *RST*, (Deterministic(13.7), .150, 0, 0), ∅, ∅, {(*ND*, 1)}), (OPb14, *ND*, (Normal(11.5, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B5]}$ = {(OPb13, *ND*, (Deterministic(18.8), 0, 0, 0), ∅, ∅, ∅), (OPb13, *RST*, (Deterministic(18.8), .200, 0, 0), ∅, ∅, {(*ND*, 1)}), (OPb14, *ND*, (Normal(11.5, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B\_R]}$ = {(OPb15, *ND*, (Deterministic(3.1), 0, 0, 0), ∅, ∅, ∅)}.

$\Delta^{cr}_{[B2345]}$ = {(OPb02, {(*, *B4DFM*, ***true***, 1.2), (*, *B2CRK_B4DFM*, ***true***, 1.2), (*, *B4DFM_J5BB*, ***true***, 1.2)})};

$\Delta^{cr}_{[B234]}$ = {(OPb04, {(*, *B4DFM*, ***true***, 1.2), (*, *B4DFM_J4TF*, ***true***, 1.2)})}.

The functions for specifying the necessity of quality test and the limit of rework times of [B] are as follows:

$$\cup v^{qt}_{[B^*]}(o \in O_{[B^*]}) = \begin{pmatrix} \textbf{\textit{false}} & \textbf{\textit{, if}}\ o = \text{OPb15} \\ \textbf{\textit{true}} & \textbf{\textit{, otherwise}} \end{pmatrix};$$

$$\cup v^{rw}_{[B^*]}(o \in O_{[B^*]}) = \begin{pmatrix} \textbf{0} & \textbf{\textit{, if}}\ o = \text{OPb14}\ \textbf{\textit{or}}\ \text{OPb15} \\ \textbf{1} & \textbf{\textit{, elseif}}\ o = \text{OPb08, OPb09,}\ \textbf{\textit{or}}\ \text{OPb11} \\ \textbf{2} & \textbf{\textit{, elseif}}\ o = \text{OPb07, OPb10, OPb12,}\ \textbf{\textit{or}}\ \text{OPb13} \\ \textbf{\textit{inf.}} & \textbf{\textit{, otherwise}} \end{pmatrix},$$

and the operation type specification function is as follows:

$$\tau^{o}_{[B^*]}(o \in O_{[B^*]}) = \begin{pmatrix} \textbf{\textit{virtual}} & \textbf{\textit{, if}}\ o = \text{OPbv1}\ \textbf{\textit{or}}\ \text{OPbv2} \\ \textbf{\textit{disassembly}} & \textbf{\textit{, elseif}}\ o = \text{OPb01, OPb02, OPb03, OPb04, OPb05,}\ \textbf{\textit{or}}\ \text{OPb06} \\ \textbf{\textit{cleaning}} & \textbf{\textit{, elseif}}\ o = \text{OPb07, OPb08, OPb11, OPb12,}\ \textbf{\textit{or}}\ \text{OPb13} \\ \textbf{\textit{machinging}} & \textbf{\textit{, elseif}}\ o = \text{OPb09}\ \textbf{\textit{or}}\ \text{OPb10} \\ \textbf{\textit{reassembly}} & \textbf{\textit{, elseif}}\ o = \text{OPb14} \\ \textbf{\textit{testing}} & \textbf{\textit{, otherwise}} \end{pmatrix}.$$

Last, the arrival information of [B] is as follows:

$\Delta^{pa}_{[B]}$ = (Poisson(800), Normal(30, 6.0), {(*ND*, .9), (*J1BA*, .1)}).

E.3. Combinational information model

The workstation allocation for each operation is as follows:

$$\lambda^{wa}(o \in \cup O_p) = \begin{pmatrix} \{[\text{WS1}]\} & \textbf{\textit{, if}}\ o = \text{OPb01, OPb02,}\ \textbf{\textit{or}}\ \text{OPb04} \\ \{[\text{WS2}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa01, OPa02, OPa03,}\ \textbf{\textit{or}}\ \text{OPb06} \\ \{[\text{WS2}],[\text{WS3}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa04, OPb03,}\ \textbf{\textit{or}}\ \text{OPb05} \\ \{[\text{WS4}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa07, OPb09,}\ \textbf{\textit{or}}\ \text{OPb10} \\ \{[\text{WS5}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa06, OPb07, OPb12,}\ \textbf{\textit{or}}\ \text{OPb13} \\ \{[\text{WS6}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa05, OPb08,}\ \textbf{\textit{or}}\ \text{OPb11} \\ \{[\text{WS7}]\} & \textbf{\textit{, elseif}}\ o = \text{OPa08}\ \textbf{\textit{or}}\ \text{OPb14} \\ \{[\text{WS8}]\} & \textbf{\textit{, otherwise}} \end{pmatrix},$$

and the resource performance for each operation is as follows:

Δ^{rp} = {([RC1_1], OPb1, (1.41, 1.07)), ([RC1_2], OPb1, (0.85, 0.73)), ([RC2_1], OPa1, (1.23, 0.80)), ([RC2_1], OPa2, (0.90, 0.87)), ([RC2_1], OPa4, (1.03, 0.92)), ([RC2_1], OPb6, (1, 0.82)), ([RC2_2], OPa1, (0.85, 0.86)), ([RC2_2], OPa2, (1.31, 0.72)), ([RC2_2], OPb6, (1.22, 1)), ([RC2_3], OPa1, (1.38, 1.15)), ([RC2_3], OPa4, (0.94, 1.15)), ([RC2_3], OPb6, (1.18, 1.21)), ([RC3_1], OPa4, (0.75, 0.67)), ([RC3_1], OPb3, (0.62, 0.74)), ([RC3_1], OPb5, (0.62, 0.74)), ([RC3_2], OPa4, (0.82, 0.73)), ([RC3_2], OPb3, (0.60, 0.84)), ([RC3_2], OPb5, (0.59, 0.83)), ([RC3_3], OPa4, (0.83, 0.70)), ([RC3_3], OPb3, (0.58, 0.82)), ([RC3_3], OPb5, (0.59, 0.83)), ([RC5_1], OPa6, (1.10, 1)), ([RC5_1], OPb7, (1.10, 1.13)), ([RC5_2], OPa6, (0.93, 0.91)), ([RC5_2], OPb7, (0.95, 0.79))}.

The batch capacity calculation functions for the example QRS are defined as the following form:

$f^{bc}_{r,oB}$: $\{np_{o1}, np_{o2}, \ldots\} \rightarrow \mathbb{B}$,

where

np_{oi} number of PDSPs which requested the operation o_i processing.

The return value indicates the possibility of the operation processing. The number of elements in the function's second parameter should exactly match with the number of elements in O^B, hence the following restriction should be held:

– $|O^B| = i$.

The batch capacity function set is as follows:

$$f^{bc}_{r,O^B}(np_{o_1}, np_{o_2}, \ldots) =$$

$$\begin{pmatrix} 0.025 \times np_{OPa07} \le 1 & ,if\ r = [\text{RC4\_1}]\ and\ O^B = \{\text{OPa07}\} \\ 0.02 \times np_{OPa06} \le 1 & ,elseif\ r = [\text{RC5\_1}]\ and\ O^B = \{\text{OPa06}\} \\ 0.04 \times np_{OPa06} \le 1 & ,elseif\ r = [\text{RC5\_2}]\ and\ O^B = \{\text{OPa06}\} \\ 0.0625 \times np_{OPb09} + 0.05 \times np_{OPa10} \le 1 & ,elseif\ r = [\text{RC4\_1}]\ and\ O^B = \{\text{OPb09, OPb10}\} \\ 0.0625 \times np_{OPb07} + 0.02 \times np_{OPa12} + 0.02 \times np_{OPa13} \le 1 & ,elseif\ r = [\text{RC5\_1}]\ and\ O^B = \{\text{OPb07, OPb12, OPb13}\} \\ 0.01 \times np_{OPb07} + 0.04 \times np_{OPa12} + 0.04 \times np_{OPa13} \le 1 & ,elseif\ r = [\text{RC5\_2}]\ and\ O^B = \{\text{OPb07, OPb12, OPb13}\} \end{pmatrix}.$$

Appendix F. IRSR graphical representation of the example QRS

This section shows the example QRS represented with the graphical notation of IRSR. Figures F.1 and F.2 show the graphical IRSR model of the remanufacturing shop and used products respectively.

The graphical notation in figure F.1 represents the buffers, workstations, and their belonging resources of the remanufacturing shop of the example QRS in chapter III. In case the belonging resources of a workstation are batch resources, the rounded B under the workstation box indicates it. But the batch capacity of each resource is not marked because of its complexity.

The graphical notation of the used product [A] and [B] of the example QRS in chapter III represents operations and their relationships (refer to figure F.2). Each operation's probability of success (POS) and operation processing times distribution are marked under its box. The marked values are the average over all possible input/output quality cases, hence the value is difference with that of the representative case in table III.4.

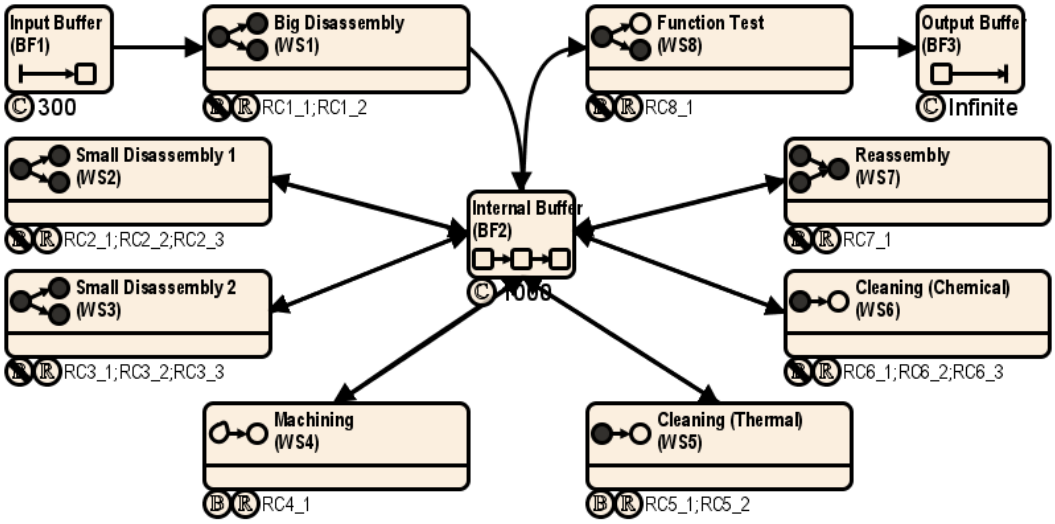


Figure F.1. IRSR graphical notation of the remanufacturing shop of the example QRS in chapter III.

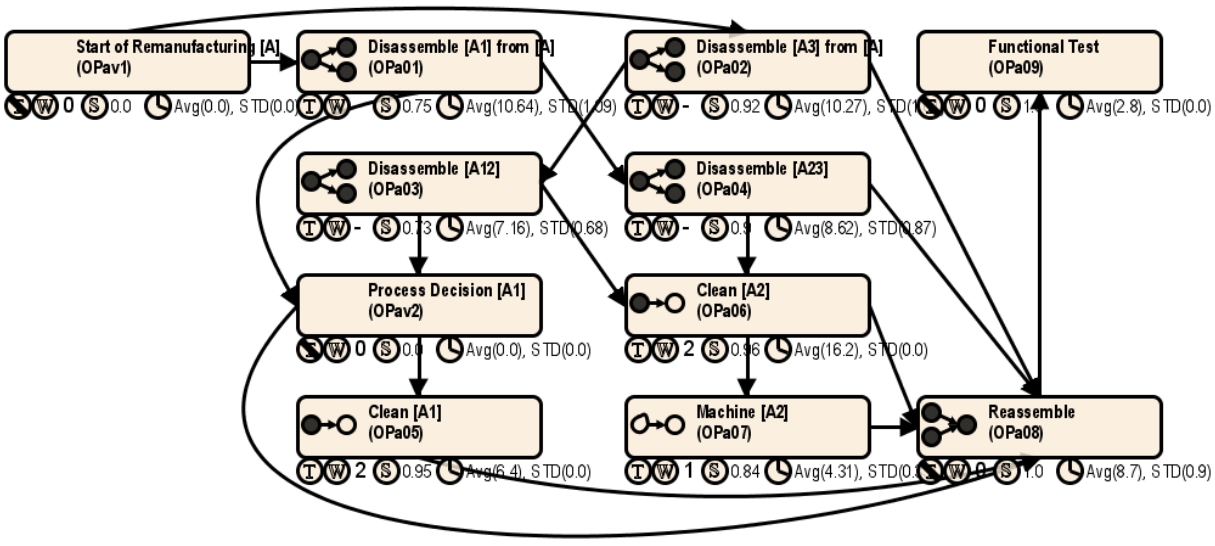


Figure F.2(a). IRSR graphical notation of the used product [A] of the example QRS in chapter III.

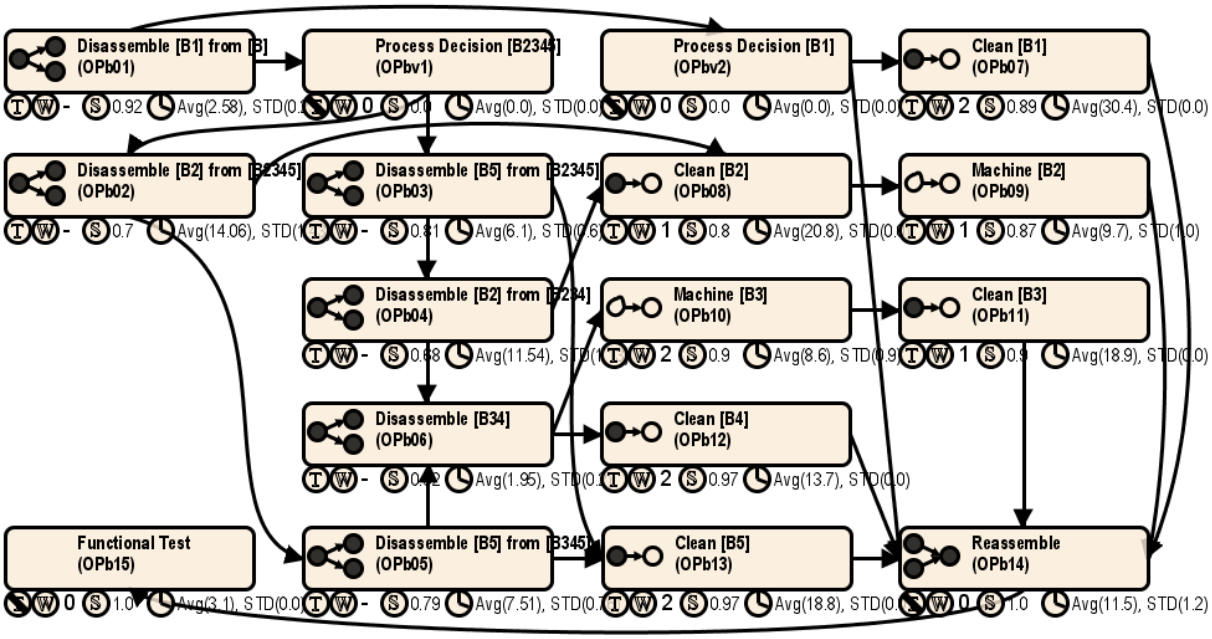


Figure F.2(b). IRSR graphical notation of the used product [A] of the example QRS in chapter III.

Appendix G. The emulated XCPN model of the example QRS

This section represents only the different part of the emulated XCPN model with IRSR model described in the above appendix E, because the other parts like operations, remanufacturing shops, and so on are exactly same.

G.1. Quality definition and statistical information of [A]

The quality of PDSPs of [A] are defined as follows:

$\boldsymbol{Q}_{[A]}$ = {*UIQ*, *DA1*, *DA3*, *RA1*, *SC*};
$\boldsymbol{Q}_{[A12]}$ = {*UIQ*, *DA1*, *DA2*, *RA1*, *SC*};
$\boldsymbol{Q}_{[A23]}$ = {*UIQ*, *DA2*, *DA3*, *SC*};
$\boldsymbol{Q}_{[Ai]}$ = {*UIQ*, *DSP*, *ND*} ($\boldsymbol{i}$ = 1, 2);
$\boldsymbol{Q}_{[A3]}$ = {*DSP*, *ND*};
$\boldsymbol{Q}_{[A\_R]}$ = {*ND*},

where

UIQ	unidentified quality;
DAi	[Ai] defect (i = 1, 2, 3);
RA1	[A1] rust;
SC	separation completion without additional defects;
DSP	defect to-be-disposed-of;
ND	no defect,

and their mapping functions are as follows:

$\boldsymbol{\lambda}^{cq}_{[A12]}(\boldsymbol{q}, \boldsymbol{p}) = \boldsymbol{\lambda}^{cq}_{[A23]}(\boldsymbol{q}, \boldsymbol{p}) = UIQ$;

$$\boldsymbol{\lambda}^{cq}_{[A1]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} UIQ & \boldsymbol{,if\ q} = RA1 \\ DSP & \boldsymbol{,elseif\ q} = DA1 \\ ND & \boldsymbol{,otherwise} \end{pmatrix};$$

$$\boldsymbol{\lambda}^{cq}_{[Ai]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} DSP & \boldsymbol{,if\ q} = DAi \\ UIQ & \boldsymbol{,otherwise} \end{pmatrix} (\boldsymbol{i} = 2, 3).$$

The operation processing statistical information of [A] is as follows:

$\boldsymbol{\Delta}^{op}_{[A]}$ = {(OPa01, *UIQ*, (Normal(12.3, 1.2), 0.009, 0, 0.388), ∅, {(*DA1*, 0.67), (*RA1*, 0.33)}, {(*SC*, 1)}),
(OPa02, *UIQ*, (Normal(11, 1.1), 0.006, 0, 0.005), ∅, {(*DA3*, 1)}, {(*SC*, 1)})};
$\boldsymbol{\Delta}^{op}_{[A12]}$ = {(OPa03, *UIQ*, (Normal(7.1, 0.7), 0.026, 0, 0.382), ∅, {(*DA1*, 0.61), (*RA1*, 0.39)}, {(*SC*, 1)})};
$\boldsymbol{\Delta}^{op}_{[A23]}$ = {(OPa04, *UIQ*, (Normal(7.6, 0.8), 0.027, 0, 0.219), ∅, {(*DA2*, 0.84), (*DA3*, 0.16)}, {(*SC*, 1)})};

$\Delta^{op}_{[A1]}$ = {(OPa05, *UIQ*, (Deterministic(6.4), 0.048, 0, 0), ∅, ∅, {(*ND*, 1)}),
(OPa08, *ND*, (Normal(8.7, 0.9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A2]}$ = {(OPa06, *UIQ*, (Deterministic(16.4), 0.034, 0, 0), ∅, ∅, {(*UIQ*, 0.34) , (*ND*, 0.66)}),
(OPa07, *UIQ*, (Normal(4.3, 0.4), 0.053, 0, 0.101), ∅, {(*DSP*, 1)}, {(*ND*, 1)}),
(OPa08, *ND*, (Normal(8.7, 0.9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A3]}$ = {(OPa08, *ND*, (Normal(8.7, 0.9), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[A\_R]}$ = {(OPa09, *ND*, (Deterministic(2.8), 0, 0, 0), ∅, ∅, ∅)}.

G.2. Quality definition and statistical information of [B]

The quality of PDSPs of [B] are defined as follows:

$\boldsymbol{Q}_{[B]}$ = {*UIQ*, *DB1*, *RB1*, *SC*};
$\boldsymbol{Q}_{[B2345]}$ = {*UIQ*, *DB2*, *DB5*, *SC*};
$\boldsymbol{Q}_{[B234]}$ = {*UIQ*, *DB2*, *SC*};
$\boldsymbol{Q}_{[B345]}$ = {*UIQ*, *DB5*, *SC*};
$\boldsymbol{Q}_{[B34]}$ = {*UIQ*, *DB3*, *DB4*, *SC*};
$\boldsymbol{Q}_{[Bi]}$ = {*UIQ*, *DSP*, *ND*} (***i*** = 1, …, 5);

where

UIQ unidentified quality;
DBi [Bi] defect (***i*** = 1, …, 5);
RB1 [B1] rust;
SC separation completion without additional defects;
DSP defect to-be-disposed-of;
ND no defect,

and their mapping functions are as follows:

$\lambda^{cq}_{[B2345]}(\boldsymbol{q}, \boldsymbol{p}) = \lambda^{cq}_{[B234]}(\boldsymbol{q}, \boldsymbol{p}) = \lambda^{cq}_{[B345]}(\boldsymbol{q}, \boldsymbol{p}) = \lambda^{cq}_{[B34]}(\boldsymbol{q}, \boldsymbol{p}) = UIQ$;

$$\lambda^{cq}_{[B1]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} UIQ & \textbf{,if } \boldsymbol{q} = RB1 \\ DSP & \textbf{,elseif } \boldsymbol{q} = DB1 \\ ND & \textbf{,otherwise} \end{pmatrix};$$

$$\lambda^{cq}_{[Bi]}(\boldsymbol{q}, \boldsymbol{p}) = \begin{pmatrix} DSP & \textbf{,if } \boldsymbol{q} = DAi \\ UIQ & \textbf{,otherwise} \end{pmatrix} (\boldsymbol{i} = 2, 3, 4, 5);$$

The operation processing statistical information of [A] is as follows:

$\Delta^{op}_{[B]}$ = {(OPb01, *UIQ*, (Normal(2.9, 0.3), 0.011, 0, 0.163), ∅, {(*DB1*, 0.43), (*RB1*, 0.57)}, {(*SC*, 1)})};

$\Delta^{op}_{[B2345]}$ = {(OPb02, *UIQ*, (Normal(14.4, 1.4), 0.194, 0, 0.023), ∅, {(*DB2*, 1)}, {(*SC*, 1)}),
(OPb03, *UIQ*, (Normal(4.8, 0.5), 0.088, 0, 0.172), ∅, {(*DB5*, 1)}, {(*SC*, 1)})};

$\Delta^{op}_{[B234]}$ = {(OPb04, *UIQ*, (Normal(11.8, 1.2), 0.147, 0, 0.018), ∅, {(*DB2*, 1)}, {(*SC*, 1)})};

$\Delta^{op}_{[B234]}$ = {(OPb05, *UIQ*, (Normal(5.9, 0.6), 0.041, 0, 0.166), ∅, {(*DB5*, 1)}, {(*SC*, 1)})};

$\Delta^{op}_{[B34]}$ = {(OPb06, *UIQ*, (Normal(2.3, 0.2), 0.1, 0, 0.14), ∅, {(*DB3*, 0.35), (*DB4*, 0.65)}, {(*SC*, 1)})};

$\Delta^{op}_{[B1]}$ = {(OPb07, *UIQ*, (Deterministic(31.5), 0.114, 0, 0), ∅, ∅, {(*ND*, 1)}),
(OPb14, *UIQ*, (Normal(12.3, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B2]}$ = {(OPb08, *UIQ*, (Deterministic(20.8), 0.18, 0, 0.012), ∅, {(*DSP*, 1)}, ∅),
(OPb09, *UIQ*, (Normal(9.7, 1), 0, 0, 0.131), ∅, {(*DSP*, 1)}, {(*ND*, 1)}),
(OPb14, *UIQ*, (Normal(12.3, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B3]}$ = {(OPb10, *UIQ*, (Normal(8.6, 0.9), 0, 0, 0.108), ∅, {(*DSP*, 1)}, ∅),
(OPb11, *UIQ*, (Deterministic(18.9), 0.097, 0, 0), ∅, ∅, {(*ND*, 1)}),
(OPb14, *UIQ*, (Normal(12.3, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B4]}$ = {(OPb12, *UIQ*, (Deterministic(13.7), 0.031, 0, 0), ∅, ∅, {(*ND*, 1)}),
(OPb14, *UIQ*, (Normal(12.3, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B5]}$ = {(OPb13, *UIQ*, (Deterministic(18.8), 0.028, 0, 0), ∅, ∅, {(*ND*, 1)}),
(OPb14, *UIQ*, (Normal(12.3, 1.2), 0, 0, 0), ∅, ∅, ∅)};

$\Delta^{op}_{[B\_R]}$ = {(OPb15, *ND*, (Deterministic(3.1), 0, 0, 0), ∅, ∅, ∅)}.

Appendix H. Calculation of the expected operation result with rework consideration

The $\boldsymbol{OQ_{o,iq}}$ (def. VIII.4.18) is constructed based on the operation statistics information and the limit of rework times. It can be gathered by recursive multiplication of the occurrence ratio. This section explain the $\boldsymbol{OQ_{o,iq}}$ generation method in detail with the cleaning operation OPa06 of the part [A2] with the *UIQ* quality of the example product [A] in chapter III (refer to section III.2.1). The quality instances of [A2] are defined above in section IV.2.1, and the output quality after the operation processing are introduced in figure IV.3.

If the rework is not considered, the output quality set for the input quality *UIQ* and the operation OPa06 $\boldsymbol{OQ^{nr}_{OPa06,SoG,0}}$ is as follows:

$\boldsymbol{OQ^{nr}_{OPa06,UIQ,0}}$ = {(*UIQ*, *false*, 0.032, 1), (*SoG*, *true*, 0.1936, 1), (*ND*, *true*, 0.7744, 1)},

where the element of each instance corresponds to the output quality, operation completion or not, and the occurrence frequency ratio. The subscript 0 in $\boldsymbol{OQ^{nr}_{OPa06,UIQ,0}}$ means 0 times rework is considered. Now to consider a rework, the UIQ quality of non-completion case should be reworked and the result set changes to as follows:

$\boldsymbol{OQ^{nr}_{OPa06,UIQ,1}}$ = {(*UIQ*, *false*, 0.001, 2), (*SoG*, *true*, 0.0062, 2), (*ND*, *true*, 0.0248, 2), (*SoG*, *true*, 0.1936, 1), (*ND*, *true*, 0.7744, 1)},

where the numbers in the first three instances are respectively calculated by 0.032×0.032, 0.032×0.1936, 0.032×0.7744. OPa06 permits maximum 2 times rework, hence the above multiplication action should be done once more for the instance (*UIQ*, *false*, 0.001024, 2), and the $\boldsymbol{OQ_{OPa06,UIQ,2}}$ is as follows:

$\boldsymbol{OQ^{nr}_{OPa06,UIQ,2}}$ = {(*UIQ*, *false*, 0, 3), (*SoG*, *true*, 0.0002, 3), (*ND*, *true*, 0.0008, 3), (*SoG*, *true*, 0.0062, 2), (*ND*, *true*, 0.0248, 2), (*SoG*, *true*, 0.1936, 1), (*ND*, *true*, 0.7744, 1)}.

To get the final result of $\boldsymbol{OQ_{OPa06,UIQ}}$, the non-completion case should be converted to the disposal quality. Only the first instance is non-completion case in the above result, hence the *UIQ* in (*UIQ*, *false*, 0, 3) should be changed to *CoG*. But the occurrence frequency ratio is 0, hence the final operation processing result set $\boldsymbol{OQ_{OPa06,UIQ}}$ is gathered by eliminating the first instance and the value indicating the operation completion as follows:

$\boldsymbol{OQ^{nr}_{OPa06,UIQ}}$ = {(*SoG*, 0.0002, 3), (*ND*, 0.0008, 3), (*SoG*, 0.0062, 2), (*ND*, 0.0248, 2), (*SoG*, 0.1936, 1), (*ND*, 0.7744, 1)}.

Appendix I. Heuristic method of different dispatching rule allocation to each workstation

The best dispatching rule of each workstation for each system state is found by the following six steps:

a. find the default dispatching rule;
b. find neighboring workstation combinations from the perspective of the visiting sequence of PDSP;
c. classify dispatching rules into some groups depending on their performance;
d. calculate the performance for each combination of dispatching rule groups for each neighboring workstation combination;
e. find the best combination of dispatching rule groups for all workstations based on the result in step 3;
f. find the best dispatching rule for each workstation.

The following consecutive subsections describe the detailed algorithm for each step of the heuristics. We omit the explanation of step a, b, and e, because they are the same with the step a, b and d in section VIII.6.3. All the unexplained symbols in the following sub-sections are defined in chapters VI and VIII.

I.1. Finding neighboring workstations (step b)

The workstation visiting sequence in the QRS is not definite because the sequence is different depending on the PDSPs. For example, the part [B2] visits the workstation [WS4] after the workstation [WS6], but on the contrary [B3] visits in a reverse sequence. It means that the workstation visiting sequence cannot be perceived simply. Therefore this thesis finds the workstation visiting sequence by the following steps:

– give a neighboring score, which is a merged value of preceding and following scores, to each pair of workstations;
– find the best neighboring combination that maximizes the sum of the neighboring score.

Because the neighboring score, preceding score, and following score should be given to each pair of workstations, this thesis handles it with a $n \times n$ matrix where n is the number of workstations. The neighboring score matrix is defined as follows:

$$NS := (scr_{i,j})_{i,j = 1, 2, \ldots, n, \text{ and } i \neq j},$$

where

$scr_{i,j} \in \mathbb{R}^+$ neighboring score of a workstation ws_i to a workstation ws_j.

The matrix NS is calculated by the sum of two matrices: preceding score matrix NS_p and following score matrix NS_f. The two matrices are defined as follows:

$$NS_p := (scr^p_{i,j})_{i,j = 1, 2, \ldots, n, \text{ and } i \neq j};$$

$NS_f := (scr^f_{i,j})_{i,j = 1, 2, \ldots, n, \text{ and } i \neq j}$,

where

$Scr^p_{i,j} \in \mathbb{R}^+$ preceding score of a workstation ws_i to a neighboring workstation ws_j;

$Scr^f_{i,j} \in \mathbb{R}^+$ following score of a workstation ws_i to a neighboring workstation ws_j.

This thesis does not count on a workstation to be a neighbor of itself, because the objective of finding workstation combinations is to examine the performance of the different dispatching rule allocation cases to different workstations. Hence the values of $scr_{i,j}$, $scr^p_{i,j}$, and $scr^f_{i,j}$ are compulsively set as null. To make the two matrices NS_p and NS_f intermediate matrices NSI_p and NSI_f are constructed first by summing the frequency of PDSP's visiting ws_j right after visiting ws_i as follows:

$NSI_p := (scrI^p_{i,j})_{i,j = 1, 2, \ldots, n, \text{ and } i \neq j}$;

$NSI_f := (scrI^f_{i,j})_{i,j = 1, 2, \ldots, n, \text{ and } i \neq j}$,

where

$$scrI^p_{i,j} = \sum_{up} \left(fr^{arr}_{up} \times \sum_{o_n, o_m \in O^I_{up}} \begin{cases} 0 & , if\ o_m \notin \lambda^{lino}(o_n), \\ & \quad ws_i \notin \lambda^{wa}(o_m), or\ ws_j \notin \lambda^{wa}(o_n) \\ 1 \Big/ \left(\left| \lambda^{wa}(o_n) \right| \times \left| \lambda^{wa}(o_m) \right| \right) & , if\ otherwise \end{cases} \right);$$

$$scrI^f_{i,j} = \sum_{up} \left(fr^{arr}_{up} \times \sum_{o_n, o_m \in O^I_{up}} \begin{cases} 0 & , if\ o_m \notin \lambda^{lino}(o_n), \\ & \quad ws_i \notin \lambda^{wa}(o_n), or\ ws_j \notin \lambda^{wa}(o_m) \\ 1 \Big/ \left(\left| \lambda^{wa}(o_n) \right| \times \left| \lambda^{wa}(o_m) \right| \right) & , else\ if\ \tau^{io}(o_n) = disassembly \\ 1 \Big/ \left(\left| \lambda^{wa}(o_n) \right| \times \left| \lambda^{wa}(o_m) \right| \times \left| \lambda^{ino}(o_n) \right| \right) & , if\ otherwise \end{cases} \right),$$

where

$up \in P^U$ product in the QRS;

$fr^{arr}_{up} = fq^{arr}_{up} \Big/ \sum_{p_u \in P_U} fq^{arr}_{p_u}$ arrival frequency ratio of product up;

$O^I_{up} = O^{ri}_{up}$ set of all operations in the all alternative processes of the product up to the end, in other words, integrated operation set which encloses all operations of PDSPs derived from up;

$$\lambda^{lino}(o) = \begin{cases} \lambda^{ino}(o) & , \lambda^{ino}(o) \neq nl. \\ \bigcup_{p \in \lambda^d(o)} o^s_p & , \lambda^{ino}(o) = nl.\ and\ \lambda^d(o) \neq nl. \end{cases}$$

function specifying a set of next operations of the operation o in O^I_{up},

where

$fq^{arr}_{up} = avg(ls_{up}) / avg(ai_{up})$ the number of arrival product per unit time;

$$O^{ri}_{p} = \begin{cases} O_p \cup \left(\bigcup_{dp \in \lambda^d(o^e{}_p)} O^{ri}{}_{dp} \right) & , \lambda^d(o^e{}_p) \neq nl. \\ O_p & , \lambda^d(o^e{}_p) = nl. \end{cases}$$

set of all the following operations of PDSP p, including operations of disassembled/reassembled PDSPs;

$\tau^{io}: \cup_{p \in P} \tau^{o}_{p}$ integrated function specifying the operation type;

$\lambda^{ino}: \cup_{p \in P} \lambda^{no}_{p}$ integrated function specifying the next operations of an operation.

$scrI^{p}_{i,j}$ and $scrI^{f}_{i,j}$ are calculated by examining each used products workstation visiting sequence. Used products' examined results are averaged by the weight, the arrival frequency fr^{arr}_{up}, to give more weight to the used products which are handled more in the remanufacturing shop. When examining a used product, all the combinations of two operations are considered. Because of the explanation efficiency, The function $scrI^{f}_{i,j}$ will be explained first for the case by case calculation.

Three cases are possible in calculation of $scrI^{f}_{i,j}$. The first condition means the operation combination is not required to be considered, because the preceding operation o_n or following operation o_m cannot be done by the workstation w_i or w_j. The other two cases are for the case w_i and w_j can process o_n and o_m, respectively. The second condition is for the case a preceding operation o_n is the disassembly operation; a disassembled PDSP sequentially does o_n and o_m. It means that o_m is definitely the following operation of o_n. Basically the score 1 should be given for the case, but o_n or o_m can be processed by multiple workstations, and the possibility of sequential visit of w_i and w_j decreases as the number of alternative workstations which can process the operations. Hence the score should be divided by the number of workstations. The last condition in the formula considers the alternative next operation of o_n in addition to the second condition. In case the operation o_n is not a disassembly operation, the operation o_m is one of o_n's next alternative operations. Hence it should be divided more by the number of next alternative operations of the preceding operation o_n, because o_m cannot be the following operation of o_n by the o_n's selecting another next operation among the candidates.

The calculation cases of $scrI^{p}_{i,j}$ is simpler than that of $scrI^{f}_{i,j}$. The first condition is same with $scrI^{f}_{i,j}$'s first condition. The second condition is for the case w_i and w_j can process o_m and o_n, respectively; the o_n is definitely the predecessor of o_m independent with the operation type of o_n. Hence it is enough to consider only the number of alternative workstations which can process the operations.

The functions λ^{ino} are defined because the next operation specification function λ^{no}_{p} in the IRSR model is defined inside of a PDSP p. For the calculation of $scrI^{p}_{i,j}$ and $scrI^{f}_{i,j}$, all the following operation of used products should be linked. To link the end operation of a PDSP to the start operation of disassembled or reassembled PDSPs, the second condition of the function λ^{lino} should be defined. O^{ri}_{p} is defined from the same reason; the operation set is only defined inside of a PDSP. By the first condition in the O^{ri}_{p} definition the operations of disassembled/reassembled PDSPs are recursively added to the O^{ri}_{p}.

Each score of $scr^{p}_{i,j}$ and $scr^{f}_{i,j}$ is calculated based on the above matrices as follows:

$$Scr^{p}_{i,j} = ScrI^{p}_{i,j} \Big/ \sum_{j} ScrI^{p}_{i,j},$$

$$Scr^{f}_{i,j} = ScrI^{f}_{i,j} \Big/ \sum_{j} ScrI^{f}_{i,j}.$$

Each workstation's preceding and following neighbor workstations can be decided, if the workstations visiting sequence which maximizes the sum of neighboring score in ***NS*** is found. The solution can be found by resolving the following integer programming problem:

Objective function: Maximize $\sum_{i}\sum_{j\neq i}(Scr_{i,j} \times nsf_{i,j})$

Conditions: $\sum_{i}\sum_{j\neq i} nsf_{i,j} = n - 1;$

$$\sum_{i} nsf_{i,j} \in \{0, 1\}, \ \sum_{j} nsf_{i,j} \in \{0, 1\};$$

$$nsf_{i,j} \in \{0, 1\};$$

$$i,j = 0, 1, \ldots, n.$$

where

$nsf_{i,j}$ value specifying the selection of the neighbor relationship between ws_i and ws_j.

In the solution of the above problem, $nsf_{i,j} = 1$ means that the workstation ws_i is preceding neighbor of the workstation ws_j and that the workstation ws_j is a following neighbor of the workstation ws_i. On the contrary, $nsf_{i,j} = 0$ means that the two workstations have no neighbor relationships.

I.2. Performance calculation of dispatching rule groups combinations for each neighboring workstations (step e)

Once dispatching rules are grouped, the performance of each combination of two ***DRG***s for each neighboring workstations should be measured. When measuring the performance for neighboring workstations ws_i and ws_j which are members of ***W***, the DR^{A} is assigned to all the other workstations. The performance is tested by simulation. Therefore for each neighboring workstation of ws_i and ws_j we can create a matrix $PI_{i,j}$ which contains the performance information of all possible combinations of dispatching rule groups as follows:

$$PI_{i,j} := (perf_{i,j,n,m})_{n,m = 1, 2, \ldots, |S_{DRG}|},$$

where

$perf_{i,j,n,m} \in \mathbb{R}^{+}$ measured performance in case of assignment of the DRG_n and DRG_m to ws_i and ws_j, respectively.

Here the n and m means the index of DRG, in other words, n^{th} element of S^{DRG} is expressed as DRG_n. Refer to the section VIII. 6.3.3 for the PDSP's dispatching priority for the case of dispatching rule group allocation.

I.3. Finding the best dispatching rule group for each workstation (step f)

The best dispatching rule group for each workstation can be decided by finding a dispatching rule group combination for all workstations which maximize the sum of every performance of dispatching rule group combinations for each neighboring workstations gathered in step c (refer to section VIII.6.3.3). It can be solved by resolving the following integer programming problem:

Objective function: maximize $\sum_i \sum_j \sum_n \sum_m \left(perf_{i,j,n,m} \times csf_{i,j,n,m} \right)$;

Conditions: $\sum_n \sum_m csf_{i,j,n,m} = 1$;

$\sum_m csf_{i,j,n,m} = 1 \Leftrightarrow \sum_n csf_{j,k,n,m} = 1$;

$csf_{i,j,n,m} \in \{0, 1\}$;

$i,j,k = 0, 1, \ldots, |W|$;

$n,m = 0, 1, \ldots, |S^{DRG}|$.

where

$csf_{i,j,n,m} \in \mathbb{R}^+$ function specifying the selected dispatching rule groups combination for workstations ws_i and ws_j.

In the solution of the above problem, $csf_{i,j,n,m} = 1$ means that the best dispatching rule groups for workstation ws_i and ws_j are n^{th} and m^{th} elements of S^{DRG} respectively.

Appendix J. Software for the QRS

J.1. Simulation with QRS simulator and controller

J.1.1. Main features and way of use

Figure J.1 shows the graphical user interface (GUI) of the developed QRS simulator and controller (QRS-S&C). Although the software is developed under the consideration of the synchronization between QRS-S&C and real-world remanufacturing systems, the synchronization part is not implemented yet. The synchronization method can be different from system to system and requires specifications and protocols in the hardware level like RFID specifications, hence this thesis leaves it as further technical research and focuses on simulations.

The developed software is composed of the following three viewers and two control bars (refer to figure J.1):

– remanufacturing shop and part/process list viewer;
– remanufacturing shop and part/process information viewer;
– log viewer;
– menu bar;
– tool bar.

The next section J.1.2 explains each viewer and control bar in detail.

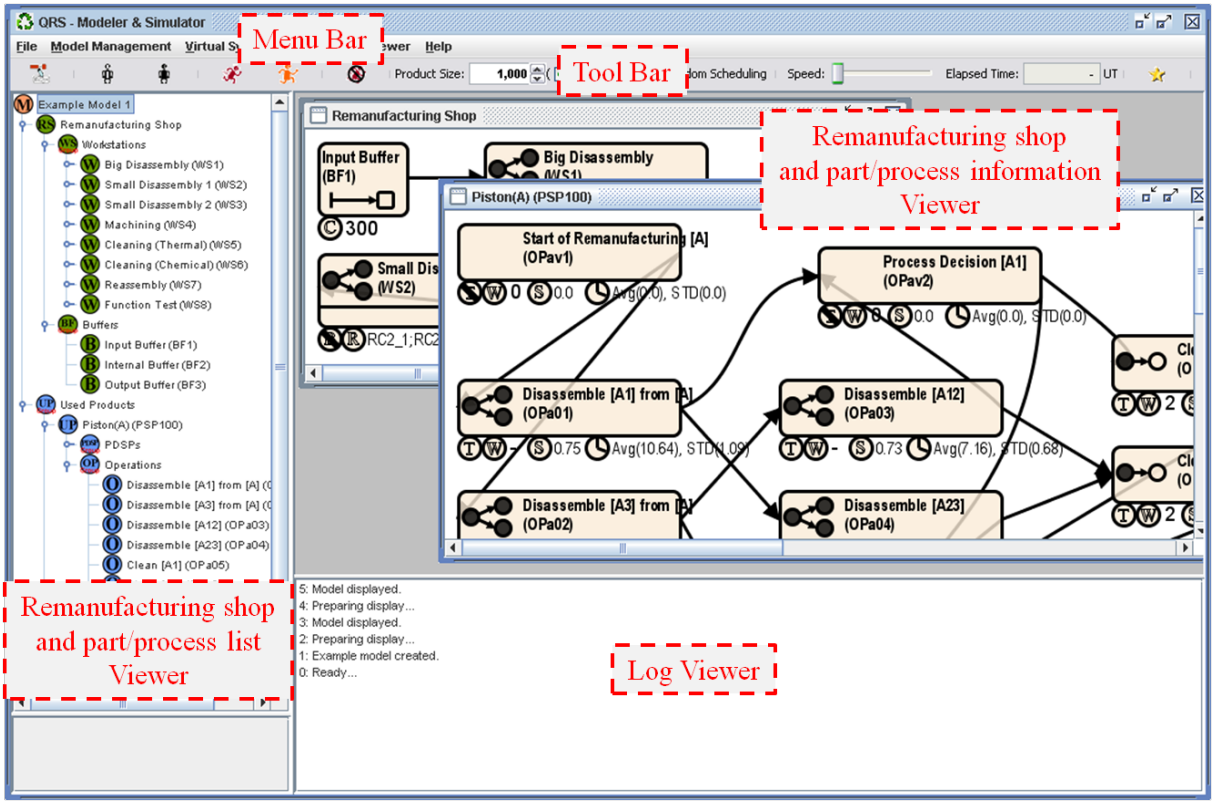


Figure J.1. Main GUI of QRS-S&C.

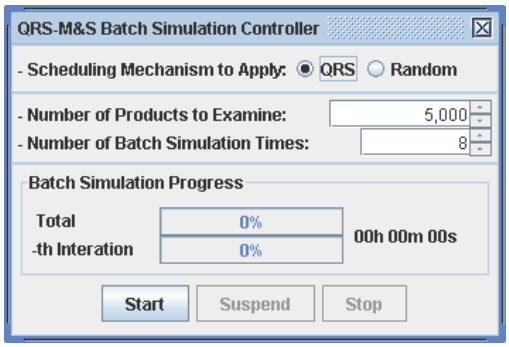


Figure J.2. GUI for batch simulation of QRS-S&C.

The simulation with QRS-S&C is done in the following sequence:

- open a QRS-S&C input file and create an IRSR model;
- create a DTPN model corresponding to the created IRSR model;
- set simulation parameters: the number of used products to be examined, used products counting criteria, and scheduling mechanism application or not;
- create virtual system;
- run a simulation;
- analyze a simulation result file.

Input and output file specifications will be discussed in section J.1.3. QRS-S&C also supports a batch simulation, which means two or more simulations under the identical simulation parameters. The virtual system composed of all the agents defined in the proposed multi-agent framework in chapter V except for the PDSP agents which are dynamically created during simulations. Each agent embeds its execution logic and the created IRSR/DTPN information.

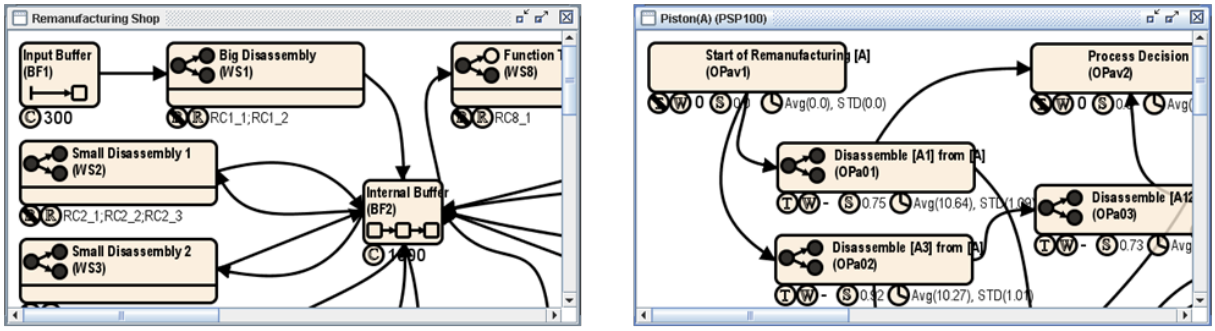
The batch simulation is done in the same way with the one-time simulation except for the omission of the virtual system creation step; the virtual system is created automatically by the software at the beginning of each simulation. Figure J.2 is the GUI for batch simulations, which can set the simulation parameters of the scheduling mechanism application or not, number of used products to be examined, and simulation times. Clicking the start button starts a batch simulation.

J.1.2. Viewers and control bars

J.1.2.1 remanufacturing shop and part/process list viewer

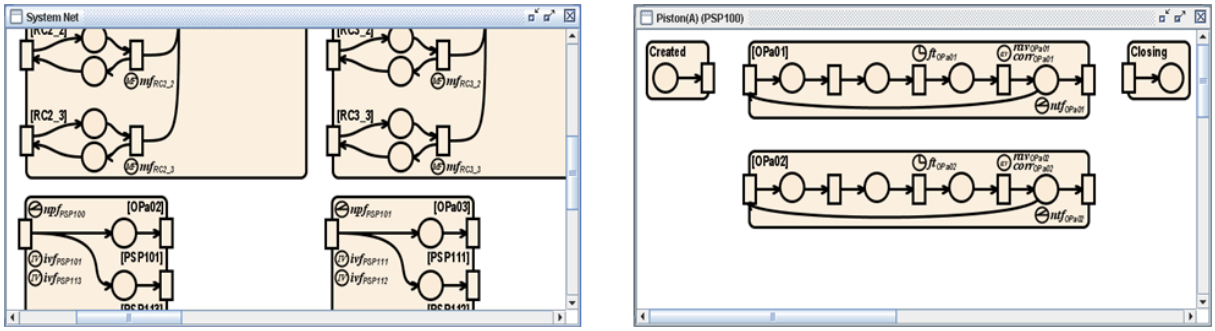
When a QRS-S&C input file is opened and a corresponding IRSR model is created, the remanufacturing shop and part/process list viewer displays the following elements in the created IRSR model:

- workstations and belonging resources in the remanufacturing shop;
- buffers in the remanufacturing shop;



(a) (b)

Figure J.3. IRSR model viewer of QRS-S&C: (a) remanufacturing shop viewer and (b) used product remanufacturing process viewer.



(a) (b)

Figure J.4. DTPN model viewer of QRS-S&C: (a) system-net viewer and (b) token-net viewer.

– used products which the remanufacturing shop handles;
– parts composing each used product;
– remanufacturing operations of each used product.

The more detailed information can be explored with the sub-viewers in the remanufacturing shop and part/process information viewer.

J.1.2.2 remanufacturing shop and part/process information viewer

The remanufacturing shop and part/process information viewer represents the graphical IRSR model of the read input file and the graphical DTPN model converted from the read IRSR model. It has the following four kinds of sub-viewers:

– remanufacturing shop viewer (refer to figure J.3(a));
– product-alternative process viewer (refer to figure J.3(b));
– system-net viewer (refer to figure J.4(a));
– token-net viewer (refer to figure J.4(b)).

Each viewer displays the information with the graphical notations defined in chapter VI (refer to figures VI.4 and VI.5). The software does not support information modification on the viewers; for example, ID and name change, statistical information change, and so on. The interactive modification of model should be implemented further for the more convenient modeling.

Table J.1. QRS-S&C menu structure.

Menu	Sub menu	description
File	Open…	Open an IRSR model file to simulate.
	Export to XML	Export an IRSR model and converted DTPN model to files in an XML format.
	Exit	Exit QRS-S&C.
Model	Convert IRSR to DTPN	Convert the opened IRSR model to a DTPN model.
Viewer	Remanufacturing Shop Viewer (IRSR)	Launch a remanufacturing shop viewer which displays the remanufacturing shop of the opened IRSR model.
	Product-Alternative Processes Viewers (IRSR)	Launch a product-alternative processes viewer which displays the remanufacturing process of a selected used product of the opened IRSR model.
	System Net Viewer (DTPN)	Launch a system-net viewer which displays the converted system-net from the opened IRSR model.
	Token Net Viewers (DTPN)	Launch a token-net viewer which displays the converted token-net of a selected PDSP from the opened IRSR model
Virtual System	Create Simulation Mode Virtual-System	Create multi-agent virtual system corresponds to the opened IRSR model.
Simulation	Batch Simulation	Launch a batch simulation window.
	Initialize Simulation	Initialize agents in the virtual-system; load statistical information required for simulation.
	Start Simulation	Start one-time simulation.
	One step Simulation	Suspend simulation before user select this menu again.
	Stop Simulation	Stop one-time simulation.
Help	Create Example Model	Create an example model.
	About QRS-S&C...	Display the version information.

J.1.2.3 Log viewer

The log viewer shows the logs related to the user actions and simulation progress; for example, opening an IRSR model, creation of a corresponding virtual system, opening viewers, numbers of arrived used products, and so on.

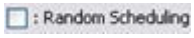
J.1.2.4 Menu bar

Table J.1 represents the menu structure of QRS-S&C and the simple explanation for each menu.

The file menu has three sub menus: open, export to XML, and exit. When the open menu is selected, QRS-S&C launches a file selection dialog that enables users to select a QRS-S&C input file which has the information of a remanufacturing system to simulate. Selection of the export to XML menu creates files which contain the created IRSR model based on the read input file and converted DTPN model in an XML format. The exit menu terminates QRS-S&C even in case the system is under simulations.

The model menu has one sub menu of convert IRSR to DTPN. Once an IRSR model is opened, it can be converted to the corresponding DTPN model. Selection of this menu executes the conversion.

Table J.2. Controls on the QRS-S&C tool bar.

GUI	Type	Description	Associated menu
	Button	Convert an IRSR model	Convert IRSR to DTPN
	Button	Create virtual-system	Create Simulation Mode Virtual-System
	Button	Initialize simulation	Initialize Simulation
	Button	Start simulation	Start Simulation
	Button	One step simulation	One step Simulation
	Button	Stop simulation	Stop Simulation
	Spinner	Set the number of to-be-handled used products	-
	Checkbox	Set if the number of to-be-handled used products is counted by the arrived used product or not	-
	Checkbox	Set if the simulation applies the PDSP's operation/resource selection scheduling mechanism or not	-
	Slider	Adjust simulation speed	-
	Textbox	Display the elapsed time during simulation	-
	Button	Launch the batch simulation window	Batch Simulation

The viewer menu has four sub menus: remanufacturing shop viewer, product-alternative processes viewers, system-net viewer, and token-net viewers. Each menu launches a viewer corresponds its menu name in the remanufacturing shop and part/process information viewer. In case a user selects the product-alternative processes viewers menu and the QRS to simulate handles two or more used products, a used product selection dialog is launched, and the product-alternative processes viewer for the selected used product is shown. In the same way, the token-net viewers menu also launches a PDSP selection window first and the token-net viewer of the selected PDSP is launched after.

The virtual system menu has one sub menu: create simulation mode virtual system. To run simulations, virtual multi-agent system should be created; selection of the create simulation model virtual system menu creates a virtual-system based on the created IRSR and converted DTPN models.

The simulation menu has five sub menus: batch simulation, initialize simulation, start simulation, one step simulation, and stop simulation. The batch simulation menu is to launch the batch simulation window (refer to figure J.2) which is explained in appendix J.1.1. To run a simulation, each agent should load required statistical information as well as set required values to some variables; for example, the used product arrival manager agent (refer to section V.2.1.3) calculates each used product's total processing time for the efficiency of its due date setting. Selection of the start simulation menu executes a simulation, and the simulation stops automatically if it reaches preset number of used products to be examined. If the one step simulation menu is selected during a simulation, QRS-S&C suspends the simulation until the start simulation or the one step simulation menu is selected again; selection of the start simulation menu continues its execution, but selection of the one step simulation menu again continues only one unit time further and suspends the simulation again.[1] Selection of the stop simulation menu directly terminates executing simulation; the compulsive simulation termination with the

[1] This menu is to monitor the system state change during simulation. But it is no use in the current version, because QRS-S&C does not support currently any monitoring tools of real-time system states.

button creates no simulation result information in the output file.

The help menu has the two kinds of sub menus: create example model and about QRS-S&C. QRS-S&C contains IRSR models for four example QRSs which are handled in this thesis: the example QRS in chapter III, the emulated XCPN model of example QRS with IRSR in section VI.5.2, the mobile phone repair shop in section VIII.7.2.1.1, and the automotive part remanufacturing shop in section VIII.7.2.1.2. Selection of each menu has the same effect with opening the IRSR model file contains the information of each remanufacturing shop by the open menu which is a sub menu of the file menu. Selection of about QRS-S&C launches a window which shows its version information.

J.1.2.5 Tool bar

Table J.2 shows the controls on the tool bar of QRS-S&C. All the button type controls are associated with the menus in the menu bar which are discussed in the above section, hence this section omits their explanation.

The simulation parameters are set with one spinner and two checkbox controls named product size, arrival, and random scheduling; each is to set respectively the number of used products to be examined, if the number of used products to be handled is counted by the arrived used product or not, and if the simulation applies the PDSP's operation/resource selection scheduling mechanism or not. This thesis examined 5,000 used products for simulations and 3,000 used products for pilot simulations; each number is set with the product size spinner. In case a user wants to simulate until the preset used products are arrived, the arrival checkbox should be checked. On the contrary case, the simulation runs until the number of remanufactured used products reaches to the preset number of used products to be examined. The random scheduling checkbox is to apply the PDSP operation/resource selection mechanism which this thesis proposed in section VIII.4. The checkbox should be checked for the cases like case studies of a mobile phone repair shop and an automotive remanufacturing shop (refer to section VIII.7.1) where PDSPs randomly selected operations/resources because of no quality information.

The slider and textbox control is used during simulations. The slider adjusts the simulation speed. When the button is dragged to the leftmost, QRS-S&C executes simulations fastest (about 100 unit time passes in 1 second during simulations, but it is different depending on the target QRS's complexity). The more to right the button is dragged, the slower the simulation runs.[1] The textbox control displays the elapsed time of a simulation in the unit time scale, which means the corresponding execution time of the target remanufacturing system in real.

J.1.3. Input file specification

Figure J.5 shows an input file which is opened with a simple text editor. Input files have a text format, and each line contains one information instance; for example, workstation definition, operation definition, and so on. Information which should be specified in an input file is represented in table J.3 with related features discussed in this thesis.

[1] This menu is also useful to monitor the system state change during simulation like the one step simulation menu, hence it is currently no use.

Table J.3. Input file specification.

Instance ID	Instance Description	Required attribute	Example	Related features in this thesis	
MNM	Model Name	Name of the model	QRS Example	-	
BFD	Buffer	Buffer ID	BF1	$\boldsymbol{B}$	(def. VI.2.10)
		Buffer Name	Input buffer	-	
		X-position in GUI	10	-	
		Y-position in GUI	10	-	
		Capacity	300	$\boldsymbol{v}^{cb}$	(def. VI.2.15)
		Waiting cost per unit time	0	$\boldsymbol{c}^{w}$	(def. VIII.2.7)
		Buffer Type	Input	$\boldsymbol{\tau}^{b}$	(def. VI.2.18)
WSD	Workstation	Workstation ID	WS1	$\boldsymbol{W}$	(def. VI.2.8)
		Workstation Name	Big Disassembly	-	
		Belonging resources' physical type	Machine	$\boldsymbol{\tau}^{br}$	(def. VI.2.17)
		X-position in GUI	160	-	
		Y-position in GUI	10	-	
		Belonging resources	RC1_1;RC1_2	$\boldsymbol{\lambda}^{ra}$	(def. VI.2.11)
				$\boldsymbol{R}$	(def. VI.2.9)
		Having batch resources' or not	*false*	$\boldsymbol{v}^{bp}$	(def. VI.2.14)
		Input buffers	BF1	$\boldsymbol{\lambda}^{ib}$	(def. VI.2.12)
		Output buffers	BF2	$\boldsymbol{\lambda}^{ob}$	(def. VI.2.13)
		Workstation functional type	Disassembly	$\boldsymbol{\tau}^{w}$	(def. VI.2.16)
PDD	Used product	Used product ID	PSP100	$\boldsymbol{P}^{U}$	(def. VI.2.20)
		Used product Name	Piston	-	
		Possible quality set	ND;A1CoW;…	$\boldsymbol{Q}_p$	(def. IV.2.1);
		Arrival interval distribution	Poi;500	$\boldsymbol{ai}_p$	(def. VI.2.36)
		Arrival lot size distribution	Nor;24.5;2.5	$\boldsymbol{ls}_p$	(def. VI.2.37)
		Arrival quality distribution	ND;0.7;J12AH;0.3	$\boldsymbol{qd}_p$	(def. VI.2.38)
		Initial buffer for arrived used products	BF1	-	
		Starting operation	OPa01	$\boldsymbol{o}^{s}_p$	(def. VI.2.26)
		Constraint on the part reassembly from the same used product	*false*	-	
		Disposal cost	0	$\boldsymbol{c}^{d}_p$	(def. VIII.2.11)
		Order arrival interval distribution	Poi;416	-	
		Order lot size distribution	Poi;7.14	-	
		Order due date	Set;1440;0.4;10080;0.6	-	
SPRD	Subassembly or part	Belonging used product	PSP100	-	
		PDSP id	PSP111	$\boldsymbol{P}$	(def. VI.2.20)
		PDSP name	A1	-	
		Possible quality set	ND;CoW;RST	$\boldsymbol{Q}_p$	(def. IV.2.1);
		Qualities to be disposed of	CoW	$\boldsymbol{QS}^{D}_p$	(def. VIII.4.23)
		To-be-generated PDSP after its disposal	PSP111	-	
		Initial quality of to-be-generated PDSP	ND	$\boldsymbol{nd}_p$	(def. VIII.4.22)
		First operation of to-be-generated PDSP	OPa08	$\boldsymbol{o}^{e}_{dp}$	(def. VI.2.27)
		Buffer to-be-generated PDSP to-be-sent	BF2	-	
		Disposal cost	95	$\boldsymbol{c}^{d}_p$	(def. VIII.2.11)
OPD	Operation	Operation ID	OPa01	$\boldsymbol{O}_p$	(def. VI.2.25)
		Belonging used product	PSP100	-	
		Operation name	Disassemble [A1]	-	
		X-position in GUI	10	-	
		Y-position in GUI	150	-	
		Quality examination after operation or not	*true*	-	
		Limit of rework times	0	$\boldsymbol{v}^{rw}_p$	(def. VI.2.33)
		Operation type	disassembly	$\boldsymbol{\tau}^{o}_p$	(def. VI.2.35)
OPR	Operation relationship	Belonging used product	PSP100	-	
		Operation to specify next operations	OPa01	-	
		Next operations	OPa04;OPav2	$\boldsymbol{\lambda}^{no}_p$	(def. VI.2.28)
		Generated PDSPs	PSP101;PSP111	$\boldsymbol{\lambda}^{d}$	(def. VI.2.23)

POI	Operation processing time and POF	Belonging used product	PSP100	-
		Operation ID	OPa01	-
		Input PDSP quality	ND	-
		Processing time	Nor;9.6;1	$pt_{o,iq}$ (def. IV.2.3)
		POF and POS	0;0;0.062	$pnc_{o,iq}$ (def. IV.2.4)
				$pnc^{d}_{o,iq}$ (def. IV.2.5)
				$pc^{d}_{o,iq}$ (def. IV.2.6)
QRR	Operation processing quality result	Belonging used product	PSP100	-
		Operation ID	OPa01	-
		Input PDSP quality	ND	-
		Quality ratio (non-completion)	A1CoW;1	$RPQ^{nc,d}_{o,iq}$ (def. IV.2.11)
		Quality ratio (completion with defects)	ND;0.8;A1RST;0.2	$RPQ^{c,d}_{o,iq}$ (def. IV.2.12)
		Quality ratio (success)	ND;1	$RPQ^{c}_{o,iq}$ (def. IV.2.13)
SRQ	Processing time and output quality correlation	Belonging used product	PSP100	-
		Operation ID	OPa01	-
		Input quality	ND;J12AH	-
		Output quality	A1CoW	-
		Operation completion or not	*true*	-
		Correlation	0.7	$cf^{c\text{-}pt}_{iq,oq,cmp}$ (def. IV.2.15)
QMP	Quality mapping between PDSPs	Belonging used product	PSP100	-
		PDSP id	PRT_A1	-
		PDSP quality	A1CoW	-
		Composing PDSP IDs	PSP102;PSP111	-
		Qualities of composing PDSPs	ND;CoW	λ^{cq} (def. VI.2.24)
OWA	Workstation allocation	Operation ID	OPa04	-
		Workstations in charge	WS2;WS3	λ^{wa} (def. VI.2.41)
ORP	Resource quality	Resource ID	RC2_1	-
		Operation ID	OPa01	-
		Processing time quality	1.23	$cf^{r\text{-}pt}_{o,r}$ (def. IV.2.17)
		POF quality	0.86	$cf^{r\text{-}POF}_{o,r}$(def. IV.2.18)
BPC	Batch constraint	Resource ID	RC4_1	-
		Simultaneously processable operations	OPb09;OPb10	O^{B} (def. VI.2.44)
		Capacity occupation ratio of each operation	0.0625;0.05	$f^{bc}_{r,O^{B}}$ (def. VI.2.45)
OCI	Operation processing cost	Operation ID	OPb01	-
		Resources in charge of the operation	RC1_1;RC1_2	-
		Processing cost per unit time	1;1	$c^{p}_{o,r}$ (def. VIII.2.5)
DCI	Delay penalty cost	Used product ID	PSP100	-
		Delay upper limit	500	f^{odp} (def. VIII.2.9)
		Linear coefficient	0.3	f^{odp} (def. VIII.2.9)
		Constant coefficient	-50	f^{odp} (def. VIII.2.9)
SNM	Scenario map	Scenario ID	SCN001	scn_i (def. VIII.6.3)
		Dispatching rule for workstations[a]	WS1:WS3;LPOST	-
KGM	Knowledge map	Parameter rage*	RoRU;MIN;0.15	sv_b (def. VIII.6.2)
		Scenario ID	SCN001	scn_i (def. VIII.6.3)

[a] dispatching rule for workstations and parameter range information can be repeated for different workstation sets and state variables respectively.

The information in an input file should be described as the sequence in table J.3, because some later information refer to former information. The column for related features in this thesis represents the symbols discussed in this thesis which are directly related to the attribute to be specified. Hence the related features for some attributes are empty; they are for the graphical display like X/Y-positions of a buffer, belonging information group specification like the belonging used product of an operation, or subscripts of symbols defined in this thesis for relationships with other attributes.

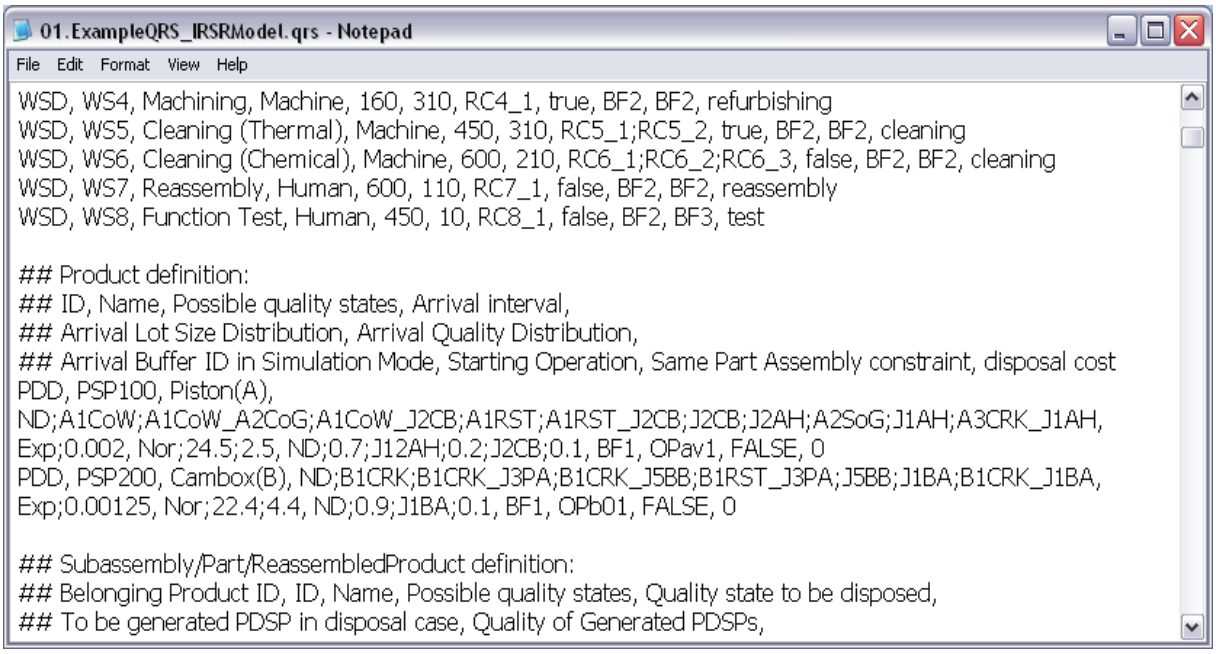


Figure J.5. Example of an input file for QRS-S&C opened with a simple text editor.

Each line of an input file corresponds to each instance; for example a buffer is defined in one line. Attributes of each instance is delimited with ‘,’ and the first attribute indicates the information type as table J.3. If an attribute have two or more values, it is delimited with ‘;’. If a value is a set, elements of the set are delimited with ‘:’ again; for example, the information on the best dispatching rule of a scenario is composed of a workstation set and a dispatching rule to be allocated, hence the workstations in the workstation set are delimited with ‘:’ as the example WS1:WS3;LPOST in table J.3.

Some information like the operation processing time and used product arrival interval indicates statistical distributions. QRS-S&C can interpret six distributions as in table J.4: deterministic, uniform, normal, exponential, Poisson, unequal discrete value set. Each can be defined with proper parameters; for example, normal distribution can be defined with its average and variance.

QRS-S&C interprets a linear delay penalty cost calculation function for each delay time range; for example, the delay penalty cost calculation function (refer to figure III.6) of the used product [A] of the example QRS in chapter III can be specified with three lines as follows:

Table J.4. Statistical distribution can be interpreted in QRS-S&C.

Distribution	Required parameters	Identifier	example
Deterministic	Constant	Det	Det;35.75
Uniform	Minimum, Maximum	Uni	Uni;800;1000
Normal	Mean, Variance	Nor	Nor;24.5;2.5
Exponential	Mean	Exp	Poi;0.002
Poisson	Mean	Poi	Poi;500
Unequal discrete value set	Constant, Occurrence ratio	Set	Set;1440;0.4;10080;0.6 *

* It means the 1,400 and 10,080 occur 40% and 60% respectively.

DCI, PSP100, 300, 0, 40
DCI, PSP100, 500, 0.3, -50
DCI, PSP100, INFINITE, 0, 100,

where 500 in the second line means the upper limit of delay time, 0.3 and -50 in the same line mean respectively the slope and constant of the linear function. The term INFINITE means the range has no upper limit. The delay penalty calculation function should be listed as the sequence of delay time ranges.

At least one dispatching rule allocation scenario should be specified in the input file as the default scenario with the identifier SCNDEF as follows:

SNM, SCNDEF, ALL;WINQ,

here the identifier ALL means the set composed of all the workstations in the target remanufacturing shop, hence QRS-S&C basically allocates WINQ for all the workstations if the current system state matches with no cases which are specified in the knowledge map. The knowledge map can be constructed in the QRS-S&C input file with the three system state variables which are discussed in section VIII.6.2: the degree of processing urgency sv^{pu}, degree of remanufacturing process progress sv^{pp}, and degree of resource utilization sv^{ru}. The identifier of each in the system is RoPU, RoPP, and RoRU respectively, and each can range from 0 to 1. The minimum and maximum value can be specified with MIN and MAX respectively. Hence the scenario SCN001 selection for the case sv^{pu} is under 0.35 and sv^{ru} is over 0.7 can be specified as follows:

KGM, RoPU;MIN;0.35, RoRU;0.7;MAX, SCN001.

The following is the complete example of an input file for the example QRS in chapter III which contains the dynamic dispatching rule allocation knowledge applied in section VIII.7.2.2.1.

```
MNM, Example QRS

BFD, BF1, Input Buffer, 10, 10, 300, 0, Input
BFD, BF2, Internal Buffer, 360, 150, 1000, 0, Internal
BFD, BF3, Output Buffer, 700, 10, Infinite, 0, Output

WSD, WS1, Big Disassembly, Machine, 160, 10, RC1_1;RC1_2, false, BF1, BF2, disassembly
WSD, WS2, Small Disassembly 1, Human, 10, 110, RC2_1;RC2_2;RC2_3, false, BF2, BF2, disassembly
WSD, WS3, Small Disassembly 2, Human, 10, 210, RC3_1;RC3_2;RC3_3, false, BF2, BF2, disassembly
WSD, WS4, Machining, Machine, 160, 310, RC4_1, true, BF2, BF2, refurbishing
WSD, WS5, Cleaning (Thermal), Machine, 450, 310, RC5_1;RC5_2, true, BF2, BF2, cleaning
WSD, WS6, Cleaning (Chemical), Machine, 600, 210, RC6_1;RC6_2;RC6_3, false, BF2, BF2, cleaning
WSD, WS7, Reassembly, Human, 600, 110, RC7_1, false, BF2, BF2, reassembly
WSD, WS8, Function Test, Human, 450, 10, RC8_1, false, BF2, BF3, test

PDD, PSP100, Piston(A), ND;A1CoW;A1CoW_A2CoG;A1CoW_J2CB;A1RST;A1RST_J2CB;J2CB;J2AH;A2SoG;J1AH;A3CRK_J1AH, Exp;0.002, Nor;24.5;2.5,
ND;0.7;J12AH;0.2;J2CB;0.1, BF1, OPav1, FALSE, 0
PDD, PSP200, Cambox(B), ND;B1CRK;B1CRK_J3PA;B1CRK_J5BB;B1RST_J3PA;J5BB;J1BA;B1CRK_J1BA, Exp;0.00125, Nor;22.4;4.4,
ND;0.9;J1BA;0.1, BF1, OPb01, FALSE, 0

SPRD, PSP100, PSP101, [A12], ND;A1CoW;A1CoW_A2SoG;A1RST;A1RST_A2SoG;A2SoG;J1AH, , , , , ,
SPRD, PSP100, PSP102, [A23], ND;A2CoG;A2SoG;A2SoG_A3CRK;A3CRK;J2AH;J2CB, , , , , ,
SPRD, PSP100, PSP111, [A1], ND;CoW;RST, CoW, PSP111, ND, OPa08, BF2, 95
SPRD, PSP100, PSP112, [A2], ND;CoG;SoG;UIQ, CoG, PSP112, ND, OPa08, BF2, 45
SPRD, PSP100, PSP113, [A3], ND;CRK, CRK, PSP113, ND, OPa08, BF2, 38
SPRD, PSP100, PSP100R, [A_R], ND, , , , , ,
SPRD, PSP200, PSP201, [B2345],
```

```
ND;B2CRK;B2CRK_B4DFM;B4DFM;B4DFM_J5BB;B5CoTA;B5CoTA_J3BA;B5CoTA_J3BA_J4TF;B5CoTA_J3PA;B5CoTA_J4TF;J3BA;J3PA;J4TF;J4TF_J5BB;J5BB;B
5RST;B5RST_J3PA;B5RST_J3PA, , , , , ,
SPRD, PSP200, PSP202, [B234],
ND;B2CRK;B2CRK_B4DFM;B3CWD;B3CWD_J4TF;B4DFM;B4DFM_J4TF;J3BA;J3BA_J4TF;J3PA;J4TF;J5BB;B4DFM_J5BB;J4TF_J5BB, , , , , ,
SPRD, PSP200, PSP203, [B345],
ND;B4DFM;B4DFM_B5CoTA;B4DFM_J5BB;B5CoTA;B5CoTA_J4TF;J4TF;J4TF_J5BB;J5BB;B5RST;J3BA;J3BA_J5BB, , , , , ,
SPRD, PSP200, PSP204, [B34], ND;B3CWD;B3CWD_B4RST;B3CWD_J4TF;B4DFM;B4DFM_J4TF;B4RST;J4TF, , , , , ,
SPRD, PSP200, PSP211, [B1], ND;CRK;RST, CRK, PSP211, ND, OPb14, BF2, 33
SPRD, PSP200, PSP212, [B2], ND;CoHW;UIQ, CoHW, PSP212, ND, OPb14, BF2, 212
SPRD, PSP200, PSP213, [B3], ND;CWD;DST, CWD, PSP213, ND, OPb14, BF2, 194
SPRD, PSP200, PSP214, [B4], ND;DFM;RST, DFM, PSP214, ND, OPb14, BF2, 27
SPRD, PSP200, PSP215, [B5], ND;CoTA;RST, CoTA, PSP215, ND, OPb14, BF2, 29
SPRD, PSP200, PSP200R, [B_R], ND, , , , , ,

OPD, OPav1, PSP100, Start of Remanufacturing [A], 10, 10, false, 0, virtual
OPD, OPa01, PSP100, Disassemble [A1] from [A], 10, 150, true, -, disassembly
OPD, OPa02, PSP100, Disassemble [A3] from [A], 10, 250, true, -, disassembly
OPD, OPa03, PSP100, Disassemble [A12], 300, 150, true, -, disassembly
OPD, OPa04, PSP100, Disassemble [A23], 300, 250, true, -, disassembly
OPD, OPav2, PSP100, Process Decision [A1], 460, 110, false, 0, virtual
OPD, OPa05, PSP100, Clean [A1], 600, 110, true, 2, cleaning
OPD, OPa06, PSP100, Clean [A2], 600, 210, true, 2, cleaning
OPD, OPa07, PSP100, Machine [A2], 600, 310, true, 1, refurbishing
OPD, OPa08, PSP100, Reassemble, 300, 410, false, 0, reassembly
OPD, OPa09, PSP100, Functional Test, 600, 410, false, 0, test
OPD, OPb01, PSP200, Disassemble [B1] from [B], 10, 0, true, -, disassembly
OPD, OPbv1, PSP200, Process Decision [B2345], 10, 100, false, 0, virtual
OPD, OPb02, PSP200, Disassemble [B2] from [B2345], 10, 200, true, -, disassembly
OPD, OPb03, PSP200, Disassemble [B5] from [B2345], 10, 300, true, -, disassembly
OPD, OPb04, PSP200, Disassemble [B2] from [B234], 10, 400, true, -, disassembly
OPD, OPb05, PSP200, Disassemble [B5] from [B345], 10, 500, true, -, disassembly
OPD, OPb06, PSP200, Disassemble [B34], 10, 600, true, -, disassembly
OPD, OPbv2, PSP200, Process Decision [B1], 400, 0, false, 0, virtual
OPD, OPb07, PSP200, Clean [B1], 400, 100, true, 2, cleaning
OPD, OPb08, PSP200, Clean [B2], 400, 200, true, 1, cleaning
OPD, OPb09, PSP200, Machine [B2], 400, 300, true, 1, refurbishing
OPD, OPb10, PSP200, Machine [B3], 400, 400, true, 2, refurbishing
OPD, OPb11, PSP200, Clean [B3], 400, 500, true, 1, cleaning
OPD, OPb12, PSP200, Clean [B4], 400, 600, true, 2, cleaning
OPD, OPb13, PSP200, Clean [B5], 400, 700, true, 2, cleaning
OPD, OPb14, PSP200, Reassemble, 790, 0, false, 0, reassembly
OPD, OPb15, PSP200, Functional Test, 790, 150, false, 0, test

OPR, PSP100, OPav1, OPa01;OPa02
OPR, PSP100, OPa01, OPa04;OPav2, PSP102;PSP111
OPR, PSP100, OPa02, OPa03;OPa08, PSP101;PSP113
OPR, PSP100, OPa03, OPav2;OPa06, PSP111;PSP112
OPR, PSP100, OPa04, OPa06;OPa08, PSP112;PSP113
OPR, PSP100, OPav2, OPa05;OPa08,, RST, OPa05
OPR, PSP100, OPa05, OPa08
OPR, PSP100, OPa06, OPa07;OPa08,, ND, OPa08
OPR, PSP100, OPa07, OPa08
OPR, PSP100, OPa08, OPa09, PSP100R
OPR, PSP200, OPb01, OPbv1;OPbv2, PSP201;PSP211
OPR, PSP200, OPbv1, OPb02;OPb03
OPR, PSP200, OPb02, OPb05;OPb08, PSP203;PSP212
OPR, PSP200, OPb03, OPb04;OPb13, PSP202;PSP215
OPR, PSP200, OPb04, OPb06;OPb08, PSP204;PSP212
OPR, PSP200, OPb05, OPb06;OPb13, PSP204;PSP215
OPR, PSP200, OPb06, OPb10;OPb12, PSP213;PSP214
OPR, PSP200, OPbv2, OPb07;OPb14, , ND, OPb14
OPR, PSP200, OPb07, OPb14
OPR, PSP200, OPb08, OPb09
OPR, PSP200, OPb09, OPb14
OPR, PSP200, OPb10, OPb11
OPR, PSP200, OPb11, OPb14
OPR, PSP200, OPb12, OPb14
OPR, PSP200, OPb13, OPb14
OPR, PSP200, OPb14, OPb15, PSP200R

POI, PSP100, OPa01, ND, Nor;9.6;1, 0;0;0.062
POI, PSP100, OPa01, J12AH, Nor;15.4;1.5, 0.042;0;0.958
POI, PSP100, OPa01, J2CB, Nor;9.6;1, 0;0;0.054
POI, PSP100, OPa02, ND, Nor;9.3;0.9, 0;0;0.098
POI, PSP100, OPa02, J12AH, Nor;16.7;1.7, 0.032;0;0.027
POI, PSP100, OPa02, J2CB, Nor;4.2;0.4, 0;0;0
POI, PSP100, OPa03, ND, Nor;6.4;0.6, 0.021;0.017;0.048
```

```
POI, PSP100, OPa03, A2SoG, Nor;6.4;0.6, 0;0;0.068
POI, PSP100, OPa03, J1AH, Nor;10.2;1, 0;0;1
POI, PSP100, OPa04, ND, Nor;8.9;0.9, 0.042;0;0.068
POI, PSP100, OPa04, A2CoG, Nor;8.9;0.9, 0;0;0.128
POI, PSP100, OPa04, J2AH, Nor;16;1.6, 0.032;0;0.027
POI, PSP100, OPa04, J2CB, Nor;6.1;0.6, 0;0;0
POI, PSP100, OPa05, RST, Det;6.4, 0.05;0;0
POI, PSP100, OPa06, UIQ, Det;16.2, 0.032;0;0
POI, PSP100, OPa06, SoG, Det;16.2, 0.071;0;0
POI, PSP100, OPa07, SoG, Nor;4.4;0.4, 0.053;0;0.104
POI, PSP100, OPa08, ND, Nor;8.7;0.9, 0;0;0
POI, PSP100, OPa09, ND, Det;2.8, 0;0;0
POI, PSP200, OPb01, ND, Nor;2.3;0.2, 0;0;0.062
POI, PSP200, OPb01, J1BA, Nor;5.1;0.5, 0.084;0.032;0.175
POI, PSP200, OPb01, B1CRK_J1BA, Nor;5.1;0.5, 0.062;0;0
POI, PSP200, OPb02, ND, Nor;12.6;1.3, 0;0.271;0
POI, PSP200, OPb02, J3PA, Nor;20.2;2, 0;0;0.352
POI, PSP200, OPb02, J5BB, Nor;12.6;1.3, 0;0.271;0
POI, PSP200, OPb02, J3BA, Nor;16.4;1.6, 0;0;0.406
POI, PSP200, OPb02, J3BA_J5BB, Nor;16.4;1.6, 0;0;0.406
POI, PSP200, OPb03, ND, Nor;6.1;0.6, 0.103;0;0
POI, PSP200, OPb03, J3PA, Nor;6.1;0.6, 0.103;0;0
POI, PSP200, OPb03, J3BA, Nor;6.1;0.6, 0.103;0;0
POI, PSP200, OPb03, J5BB, Nor;6.1;0.6, 0;0;1
POI, PSP200, OPb03, J3BA_J5BB, Nor;11;1.1, 0;0;1
POI, PSP200, OPb04, ND, Nor;10.3;1, 0;0.214;0.084
POI, PSP200, OPb04, J3BA, Nor;16.5;1.7, 0;0;0.447
POI, PSP200, OPb04, J3BA_J4TF, Nor;16.5;1.7, 0;0;0.447
POI, PSP200, OPb04, J3PA, Nor;13.4;1.3, 0;0;0.387
POI, PSP200, OPb04, J4TF, Nor;10.3;1, 0;0.214;0.084
POI, PSP200, OPb05, ND, Nor;6.9;0.7, 0.037;0.026;0.059
POI, PSP200, OPb05, B4DFM, Nor;6.9;0.7, 0.037;0.026;0.059
POI, PSP200, OPb05, B4DFM_J5BB, Nor;12.4;1.2, 0;0;1
POI, PSP200, OPb05, J4TF, Nor;6.9;0.7, 0;0.02;0.038
POI, PSP200, OPb05, J4TF_J5BB, Nor;12.4;1.2, 0;0;1
POI, PSP200, OPb05, J5BB, Nor;12.4;1.2, 0;0;1
POI, PSP200, OPb06, ND, Nor;1.9;0.2, 0;0;0.034
POI, PSP200, OPb06, B3CWD, Nor;1.9;0.2, 0;0;0
POI, PSP200, OPb06, B3CWD_J4TF, Nor;2.4;0.2, 0.51;0;0
POI, PSP200, OPb06, B4DFM, Nor;1.9;0.2, 0;0;0
POI, PSP200, OPb06, B4DFM_J4TF, Nor;2.4;0.2, 0.51;0;0
POI, PSP200, OPb06, J4TF, Nor;2.4;0.2, 0.51;0;0
POI, PSP200, OPb07, RST, Det;30.4, 0.105;0;0
POI, PSP200, OPb08, UIQ, Det;20.8, 0.183;0;0.014
POI, PSP200, OPb09, ND, Nor;9.7;1, 0;0;0.13
POI, PSP200, OPb10, ND, Nor;8.6;0.9, 0;0;0.105
POI, PSP200, OPb11, DST, Det;18.9, 0.1;0;0
POI, PSP200, OPb12, ND, Det;13.7, 0;0;0
POI, PSP200, OPb12, RST, Det;13.7, 0.15;0;0
POI, PSP200, OPb13, ND, Det;18.8, 0;0;0
POI, PSP200, OPb13, RST, Det;18.8, 0.2;0;0
POI, PSP200, OPb14, ND, Nor;11.5;1.2, 0;0;0
POI, PSP200, OPb15, ND, Det;3.1, 0;0;0

QRR, PSP100, OPa01, ND, , A1CoW;1, ND;0.8;A1RST;0.2
QRR, PSP100, OPa01, J12AH, , A1CoW_A2CoG;1,
QRR, PSP100, OPa01, J2CB, , A1CoW_J2CB;1, J2CB;0.8;A1RST_J2CB;0.2
QRR, PSP100, OPa02, ND, , A2SoG;1,
QRR, PSP100, OPa02, J12AH, , A3CRK_J1AH;1, J1AH;1
QRR, PSP100, OPa02, J2CB, , , ND;1
QRR, PSP100, OPa03, ND, A2SoG;1, A1CoW;1, ND;0.8;A1RST;0.2
QRR, PSP100, OPa03, A2SoG, , A1CoW_A2SoG;1, A2SoG;0.8;A1RST_A2SoG;0.2
QRR, PSP100, OPa03, J1AH, , A1CoW_A2SoG;1,
QRR, PSP100, OPa04, ND, , A2SoG;0.5;A3CRK;0.5,
QRR, PSP100, OPa04, A2CoG, , A2SoG_A3CRK;1, A2CoG;1
QRR, PSP100, OPa04, J2AH, , A3CRK;0.4;A2SoG_A3CRK;0.6, ND;1
QRR, PSP100, OPa04, J2CB, , , ND;1
QRR, PSP100, OPa05, RST, , , ND;1
QRR, PSP100, OPa06, UIQ, , , ND;0.8;SoG;0.2
QRR, PSP100, OPa06, SoG, , , SoG;1
QRR, PSP100, OPa07, SoG, , CoG;1, ND;1
QRR, PSP100, OPa08, ND, , ,
QRR, PSP100, OPa09, ND, , ,
QRR, PSP200, OPb01, ND, , B1CRK;0.8;B1CRK_J3PA;0.1;B1CRK_J5BB;0.1, ND;0.8;B1RST_J3PA;0.1;J5BB;0.1
QRR, PSP200, OPb01, J1BA, B1CRK_J1BA;1, B1CRK;0.8;B1CRK_J3PA;0.1;B1CRK_J5BB;0.1, ND;0.8;B1RST_J3PA;0.1;J5BB;0.1
QRR, PSP200, OPb01, B1CRK_J1BA, , , B1CRK;0.8;B1CRK_J3PA;0.1;B1CRK_J5BB;0.1
QRR, PSP200, OPb02, ND, J3BA;1, , ND;0.8;J4TF;0.2
```

```
QRR, PSP200, OPb02, J3PA, , B4DFM:0.3;B2CRK:0.5;B2CRK_B4DFM:0.2, ND:0.6;J4TF:0.4
QRR, PSP200, OPb02, J5BB, J3BA_J5BB:1, , J5BB:0.8;J4TF_J5BB:0.2
QRR, PSP200, OPb02, J3BA, , B4DFM:1, ND:0.8;J4TF:0.2
QRR, PSP200, OPb02, J3BA_J5BB, , B4DFM_J5BB:1, J5BB:0.8;J4TF_J5BB:0.2
QRR, PSP200, OPb03, ND, , , ND:0.8;B5CoTA:0.1;B5RST:0.1
QRR, PSP200, OPb03, J3PA, , , J3PA:0.8;B5CoTA_J3PA:0.1;B5RST_J3PA:0.1
QRR, PSP200, OPb03, J3BA, , , J3BA:0.8;B5CoTA_J3BA:0.1;B5RST_J3BA:0.1
QRR, PSP200, OPb03, J5BB, , B5CoTA:0.8;B5CoTA_J4TF:0.2,
QRR, PSP200, OPb03, J3BA_J5BB, , B5CoTA_J3BA:0.8;B5CoTA_J3BA_J4TF:0.2,
QRR, PSP200, OPb04, ND, J3BA:1, B4DFM:1, ND:0.9;B3CWD:0.1
QRR, PSP200, OPb04, J3BA, , B4DFM:1, ND:0.9;B3CWD:0.1
QRR, PSP200, OPb04, J3BA_J4TF, , B4DFM_J4TF:1, ND:0.9;B3CWD:0.1
QRR, PSP200, OPb04, J3PA, , B4DFM:0.3;B2CRK:0.5;B2CRK_B4DFM:0.2, ND:0.9;B3CWD:0.1
QRR, PSP200, OPb04, J4TF, J3BA:1, B4DFM_J4TF:1, J4TF:0.9;B3CWD_J4TF:0.1
QRR, PSP200, OPb05, ND, J5BB:1, B5CoTA:1, ND:0.8;B5RST:0.2
QRR, PSP200, OPb05, B4DFM, B4DFM_J5BB:1, B4DFM_B5CoTA:1, ND:0.8;B5RST:0.2
QRR, PSP200, OPb05, B4DFM_J5BB, , B4DFM_B5CoTA:1, B4DFM:1
QRR, PSP200, OPb05, J4TF, J4TF_J5BB:1, B5CoTA_J4TF:1,
QRR, PSP200, OPb05, J4TF_J5BB, , B5CoTA_J4TF:1, J4TF:1
QRR, PSP200, OPb05, J5BB, , B5CoTA:1, ND:1
QRR, PSP200, OPb06, ND, , B3CWD:1, ND:0.8;B4RST:0.2
QRR, PSP200, OPb06, B3CWD, , , B3CWD:0.8;B3CWD_B4RST:0.2
QRR, PSP200, OPb06, B3CWD_J4TF, , , B3CWD:0.8;B3CWD_B4RST:0.2
QRR, PSP200, OPb06, B4DFM, , ,
QRR, PSP200, OPb06, B4DFM_J4TF, , , B4DFM:1
QRR, PSP200, OPb06, J4TF, , , ND:0.8;B4RST:0.2
QRR, PSP200, OPb07, RST, , , ND:1
QRR, PSP200, OPb08, UIQ, , CoHW:1, ND:1
QRR, PSP200, OPb09, ND, , CoHW:1,
QRR, PSP200, OPb10, ND, , CWD:1, DST:1
QRR, PSP200, OPb11, DST, , , ND:1
QRR, PSP200, OPb12, ND, , ,
QRR, PSP200, OPb12, RST, , , ND:1
QRR, PSP200, OPb13, ND, , ,
QRR, PSP200, OPb13, RST, , , ND:1
QRR, PSP200, OPb14, ND, , ,
QRR, PSP200, OPb15, ND, , ,

SRQ, PSP100, OPa01, APQ, APQ, FALSE, 1.3
SRQ, PSP100, OPa01, ND;J12AH, A1CoW, TRUE, 0.7
SRQ, PSP100, OPa07, SoG, CoG, TRUE:FALSE, 0.8
SRQ, PSP200, OPb04, APQ, B4DFM:B4DFM_J4TF, TRUE, 1.2
SRQ, PSP200, OPb02, APQ, B4DFM:B2CRK_B4DFM:B4DFM_J5BB, TRUE, 1.2

QMP, PSP100, PSP100, ND, PSP101:PSP102:PSP111:PSP113, ND:ND:ND:ND
QMP, PSP100, PSP100, A1CoW, PSP101:PSP102:PSP111:PSP113, -:ND:CoW:-
QMP, PSP100, PSP100, A1CoW_A2CoG, PSP101:PSP102:PSP111:PSP113, -:A2CoG:CoW:-
QMP, PSP100, PSP100, A1CoW_J2CB, PSP101:PSP102:PSP111:PSP113, -:J2CB:CoW:-
QMP, PSP100, PSP100, A1RST, PSP101:PSP102:PSP111:PSP113, -:ND:RST:-
QMP, PSP100, PSP100, A1RST_J2CB, PSP101:PSP102:PSP111:PSP113, -:J2CB:RST:-
QMP, PSP100, PSP100, J2CB, PSP101:PSP102:PSP111:PSP113, -:J2CB:ND:-
QMP, PSP100, PSP100, J2AH, PSP101:PSP102:PSP111:PSP113, -:J2AH:ND:-
QMP, PSP100, PSP100, A2SoG, PSP101:PSP102:PSP111:PSP113, A2SoG:-:-:ND
QMP, PSP100, PSP100, J1AH, PSP101:PSP102:PSP111:PSP113, J1AH:-:-:ND
QMP, PSP100, PSP100, A3CRK_J1AH, PSP101:PSP102:PSP111:PSP113, J1AH:-:-:CRK
QMP, PSP100, PSP101, ND, PSP111:PSP112, ND:UIQ
QMP, PSP100, PSP101, A1CoW, PSP111:PSP112, CoW:UIQ
QMP, PSP100, PSP101, A1CoW_A2SoG, PSP111:PSP112, CoW:SoG
QMP, PSP100, PSP101, A1RST, PSP111:PSP112, RST:UIQ
QMP, PSP100, PSP101, A1RST_A2SoG, PSP111:PSP112, RST:SoG
QMP, PSP100, PSP101, A2SoG, PSP111:PSP112, ND:SoG
QMP, PSP100, PSP102, ND, PSP112:PSP113, UIQ:ND
QMP, PSP100, PSP102, A2CoG, PSP112:PSP113, CoG:ND
QMP, PSP100, PSP102, A2SoG_A3CRK, PSP112:PSP113, SoG:CRK
QMP, PSP100, PSP102, A2SoG, PSP112:PSP113, SoG:ND
QMP, PSP100, PSP102, A3CRK, PSP112:PSP113, UIQ:CRK
QMP, PSP200, PSP200, ND, PSP201:PSP211, ND:ND
QMP, PSP200, PSP200, B1CRK, PSP201:PSP211, ND:CRK
QMP, PSP200, PSP200, B1CRK_J3PA, PSP201:PSP211, J3PA:CRK
QMP, PSP200, PSP200, B1CRK_J5BB, PSP201:PSP211, J5BB:CRK
QMP, PSP200, PSP200, B1RST_J3PA, PSP201:PSP211, J3PA:RST
QMP, PSP200, PSP200, J5BB, PSP201:PSP211, J5BB:ND
QMP, PSP200, PSP201, ND, PSP202:PSP203:PSP212:PSP215, ND:ND:UIQ:ND
QMP, PSP200, PSP201, B2CRK, PSP202:PSP203:PSP212:PSP215, -:ND:CoHW:-
QMP, PSP200, PSP201, B2CRK_B4DFM, PSP202:PSP203:PSP212:PSP215, -:B4DFM:CoHW:-
QMP, PSP200, PSP201, B4DFM, PSP202:PSP203:PSP212:PSP215, -:B4DFM:UIQ:-
QMP, PSP200, PSP201, B4DFM_J5BB, PSP202:PSP203:PSP212:PSP215, -:B4DFM_J5BB:UIQ:-
```

```
QMP, PSP200, PSP201, J4TF, PSP202;PSP203;PSP212;PSP215, -;J4TF;UIQ;-
QMP, PSP200, PSP201, J4TF_J5BB, PSP202;PSP203;PSP212;PSP215, -;J4TF_J5BB;UIQ;-
QMP, PSP200, PSP201, J5BB, PSP202;PSP203;PSP212;PSP215, -;J5BB;UIQ;-
QMP, PSP200, PSP201, B5CoTA, PSP202;PSP203;PSP212;PSP215, ND;-;-;CoTA
QMP, PSP200, PSP201, B5CoTA_J3BA, PSP202;PSP203;PSP212;PSP215, J3BA;-;-;CoTA
QMP, PSP200, PSP201, B5CoTA_J3BA_J4TF, PSP202;PSP203;PSP212;PSP215, J3BA_J4TF;-;-;CoTA
QMP, PSP200, PSP201, B5CoTA_J3PA, PSP202;PSP203;PSP212;PSP215, J3PA;-;-;CoTA
QMP, PSP200, PSP201, B5CoTA_J4TF, PSP202;PSP203;PSP212;PSP215, J4TF;-;-;CoTA
QMP, PSP200, PSP201, J3BA, PSP202;PSP203;PSP212;PSP215, J3BA;-;-;ND
QMP, PSP200, PSP201, J3PA, PSP202;PSP203;PSP212;PSP215, J3PA;-;-;ND
QMP, PSP200, PSP201, B5RST, PSP202;PSP203;PSP212;PSP215, ND;-;-;RST
QMP, PSP200, PSP201, B5RST_J3PA, PSP202;PSP203;PSP212;PSP215, J3PA;-;-;RST
QMP, PSP200, PSP201, B5RST_J3BA, PSP202;PSP203;PSP212;PSP215, J3BA;-;-;RST
QMP, PSP200, PSP202, ND, PSP204;PSP212, ND;UIQ
QMP, PSP200, PSP202, B2CRK, PSP204;PSP212, ND;CoHW
QMP, PSP200, PSP202, B2CRK_B4DFM, PSP204;PSP212, B4DFM;CoHW
QMP, PSP200, PSP202, B4DFM, PSP204;PSP212, B4DFM;UIQ
QMP, PSP200, PSP202, B4DFM_J4TF, PSP204;PSP212, B4DFM_J4TF;UIQ
QMP, PSP200, PSP202, J4TF, PSP204;PSP212, J4TF;UIQ
QMP, PSP200, PSP202, B3CWD, PSP204;PSP212, B3CWD;UIQ
QMP, PSP200, PSP202, B3CWD_J4TF, PSP204;PSP212, B3CWD_J4TF;UIQ
QMP, PSP200, PSP203, ND, PSP204;PSP215, ND;ND
QMP, PSP200, PSP203, B4DFM, PSP204;PSP215, B4DFM;ND
QMP, PSP200, PSP203, B4DFM_B5CoTA, PSP204;PSP215, B4DFM;CoTA
QMP, PSP200, PSP203, B5CoTA, PSP204;PSP215, ND;CoTA
QMP, PSP200, PSP203, B5CoTA_J4TF, PSP204;PSP215, J4TF;CoTA
QMP, PSP200, PSP203, B5RST, PSP204;PSP215, ND;RST
QMP, PSP200, PSP203, J4TF, PSP204;PSP215, J4TF;ND
QMP, PSP200, PSP204, ND, PSP213;PSP214, ND;ND
QMP, PSP200, PSP204, B3CWD, PSP213;PSP214, CWD;ND
QMP, PSP200, PSP204, B3CWD_B4RST, PSP213;PSP214, CWD;RST
QMP, PSP200, PSP204, B4DFM, PSP213;PSP214, ND;DFM
QMP, PSP200, PSP204, B4RST, PSP213;PSP214, ND;RST

OWA, OPa01, WS2
OWA, OPa02, WS2
OWA, OPa03, WS2
OWA, OPa04, WS2;WS3
OWA, OPa05, WS6
OWA, OPa06, WS5
OWA, OPa07, WS4
OWA, OPa08, WS7
OWA, OPa09, WS8
OWA, OPb01, WS1
OWA, OPb02, WS1
OWA, OPb03, WS2;WS3
OWA, OPb04, WS1
OWA, OPb05, WS2;WS3
OWA, OPb06, WS2
OWA, OPb07, WS5
OWA, OPb08, WS6
OWA, OPb09, WS4
OWA, OPb10, WS4
OWA, OPb11, WS6
OWA, OPb12, WS5
OWA, OPb13, WS5
OWA, OPb14, WS7
OWA, OPb15, WS8

ORP, RC2_1, OPa01, 1.23, 0.80
ORP, RC2_2, OPa01, 0.85, 0.86
ORP, RC2_3, OPa01, 1.38, 1.15
ORP, RC2_1, OPa02, 0.90, 0.87
ORP, RC2_2, OPa02, 1.31, 0.72
ORP, RC2_1, OPa04, 1.03, 0.92
ORP, RC2_3, OPa04, 0.94, 1.15
ORP, RC3_1, OPa04, 0.75, 0.67
ORP, RC3_2, OPa04, 0.82, 0.73
ORP, RC3_3, OPa04, 0.83, 0.70
ORP, RC5_1, OPa06, 1.10
ORP, RC5_2, OPa06, 0.93, 0.91
ORP, RC1_1, OPb01, 1.41, 1.07
ORP, RC1_2, OPb01, 0.85, 0.73
ORP, RC3_1, OPb03, 0.62, 0.74
ORP, RC3_2, OPb03, 0.60, 0.84
ORP, RC3_3, OPb03, 0.58, 0.82
ORP, RC3_1, OPb05, 0.62, 0.74
```

```
ORP, RC3_2, OPb05, 0.59, 0.83
ORP, RC3_3, OPb05, 0.59, 0.83
ORP, RC2_1, OPb06, , 0.82
ORP, RC2_2, OPb06, 1.22,
ORP, RC2_3, OPb06, 1.18, 1.21
ORP, RC5_1, OPb07, 1.10, 1.13
ORP, RC5_2, OPb07, 0.95, 0.79

BPC, RC4_1, OPa07, 0.025
BPC, RC5_1, OPa06, 0.02
BPC, RC5_2, OPa06, 0.04
BPC, RC4_1, OPb09;OPb10, 0.0625;0.05
BPC, RC5_1, OPb07;OPb12;OPb13, 0.0625;0.02;0.02
BPC, RC5_2, OPb07;OPb12;OPb13, 0.1;0.04;0.04

OCI, OPa01, RC2_1;RC2_2;RC2_3, 2;2;2
OCI, OPa02, RC2_1;RC2_2;RC2_3, 2;2;2
OCI, OPa03, RC2_1;RC2_2;RC2_3, 2;2;2
OCI, OPa04, RC2_1;RC2_2;RC2_3;RC3_1;RC3_2;RC3_3, 2;2;2;3.5;3.5;3.5
OCI, OPa05, RC6_1;RC6_2;RC6_3, 3;3;3
OCI, OPa06, RC5_1;RC5_2, 3;3
OCI, OPa07, RC4_1, 2.5
OCI, OPa08, RC7_1, 2
OCI, OPa09, RC8_1, 1
OCI, OPb01, RC1_1;RC1_2, 1;1
OCI, OPb02, RC1_1;RC1_2, 1;1
OCI, OPb03, RC2_1;RC2_2;RC2_3;RC3_1;RC3_2;RC3_3, 2;2;2;3.5;3.5;3.5
OCI, OPb04, RC1_1;RC1_2, 1;1
OCI, OPb05, RC2_1;RC2_2;RC2_3;RC3_1;RC3_2;RC3_3, 2;2;2;3.5;3.5;3.5
OCI, OPb06, RC2_1;RC2_2;RC2_3, 2;2;2
OCI, OPb07, RC5_1;RC5_2, 3;3
OCI, OPb08, RC6_1;RC6_2;RC6_3, 3;3;3
OCI, OPb09, RC4_1, 2.5
OCI, OPb10, RC4_1, 2.5
OCI, OPb11, RC6_1;RC6_2;RC6_3, 3;3;3
OCI, OPb12, RC5_1;RC5_2, 3;3
OCI, OPb13, RC5_1;RC5_2, 3;3
OCI, OPb14, RC7_1, 2
OCI, OPb15, RC8_1, 1

DCI, PSP100, 300, 0, 40
DCI, PSP100, 500, 0.3, -50
DCI, PSP100, INFINITE, 0, 100
DCI, PSP200, 200, 0, 100
DCI, PSP200, 500, 0.05, 90
DCI, PSP200, 1000, 0.27, -20
DCI, PSP200, INFINITE, 0, 250

SNM, SCNDEF, ALL;WINQ
SNM, SCN001, WS1:WS2:WS3;LPOST
SNM, SCN002, WS1:WS2:WS3;LPOSU, WS4:WS5:WS6;LPOSU
SNM, SCN003, WS4:WS5:WS6;MDD

KGM, RoRU;MIN;0.15, SCN001
KGM, RoRU;0.15;0.3, SCN002
KGM, RoRU;0.3;0.45, SCN003.
```

J.1.4. Output file specification

Figure J.6 shows an example of a simulation result output file of QRS-S&C, where values are delimited with the tab character. It contains the each simulation's system performances with conventional measures and the proposed cost based measure as follows:

- flow time measures: mean, maximum, STD;
- tardiness measures: mean, maximum, STD, percentage of tardy jobs;
- lateness measures: mean, STD;
- work in process measures: makespan, throughput;

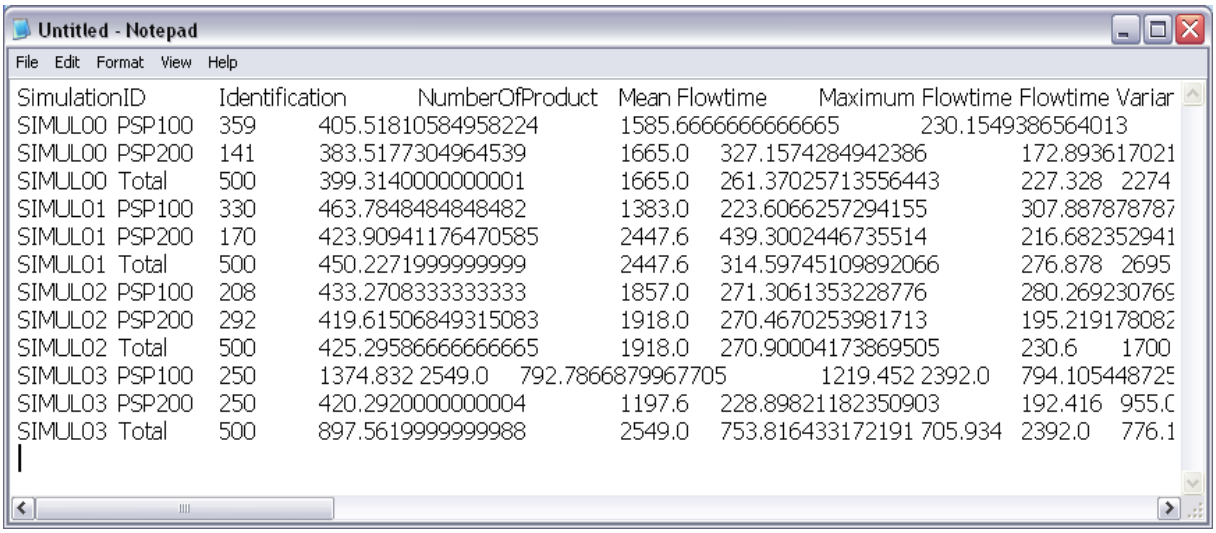


Figure J.6. Example of an output file for QRS-S&C opened with a simple text editor.

– cost measures: operation cost, waiting cost, delay cost, disposal cost, total remanufacturing cost.

The performances are calculated for each used product as well as for the whole system; the belonging used products of result records are distinguished by the identification field. System performances of different QRS conditions can be examined by comparing the above measures from simulations for different conditions; for example, to compare performance difference between EDD, FCFS, WINQ and SPT dispatching rules application cases, we can simulate with four different input files contain following four kinds different default scenario information respectively:

– SNM, SCNDEF, ALL;EDD;
– SNM, SCNDEF, ALL;FCFS;
– SNM, SCNDEF, ALL;WINQ;
– SNM, SCNDEF, ALL;SPT.

The rest of the input files except for the above default scenario part should be exactly the same with each other for the correct comparison. Different dispatching rule allocation knowledge can be compared as the same way with the dispatching rule comparison.

J.2. Structure of the QRS controller and simulator

The composing class packages of QRS-S&C and their roles are as follows:

– mainManager_GUI:main window launching, user interactive control, file input/output and log handling;
– QRSModel: IRSR and DTPN model handling, model conversion, model analysis like total processing time calculation;
– common: statistical variable generation, dispatching rule handling, mathematical calculation;
– agentFramework: agent system creation, simulation time handling, autonomous agent's self handling.

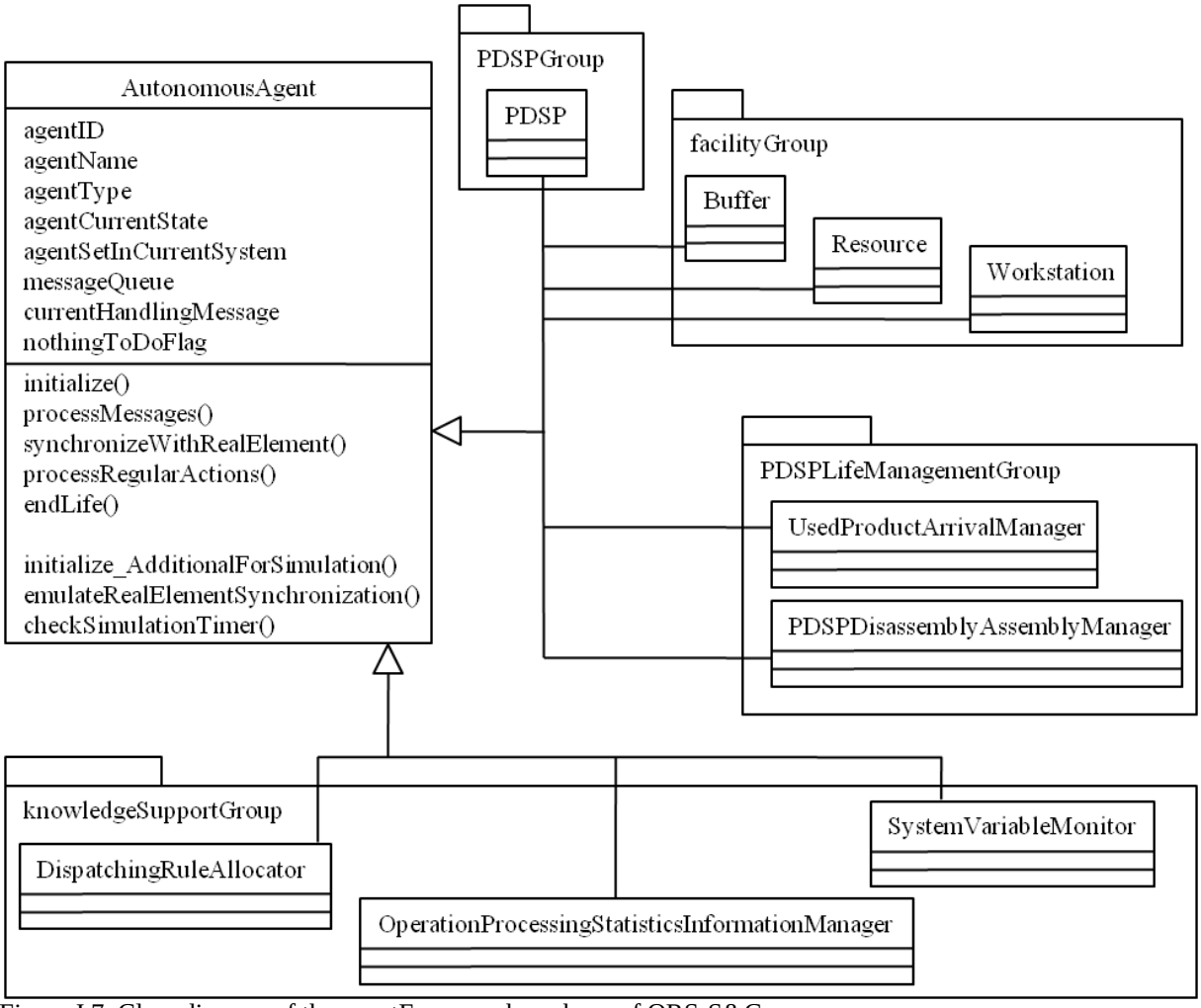


Figure J.7. Class diagram of the agentFramework package of QRS-S&C.

This section explains the overall structure of the classes only in the agentFramework package with focus on the internal structure of agents, because the classes in the other packages are not so important for the contents in this thesis but have many data management and event handling issues in the programming level. QRS-S&C is developed on the Java platform with the development tool of Eclipse SDK (version 3.4.2). It is a standalone system, hence its communication protocol parts should be modified to utilize it for the distributed system.

Figure J.7 represents the classes for the multi-agent framework in the agentFramework package of QRS-S&C, where the AutonomousAgent class which extends the thread class is defined as the abstract parent class and all the other agent classes inherit it and implement defined functions. An object created based on a class in the figure J.7 corresponds to the agents having the same name in the proposed multi-agent structure (refer to figure V.4). Although the package contains many other classes for data handling and AutonomousAgent has many other attributes and methods, this section does not discuss them as the same reason with omitting the explanation of other packages. The last four methods of AutonomousAgent in figure J.7 are required only when the QRS-S&C is executed for simulation.

The simplified main routine of AutonomousAgent is as follows:

```
while (no terminate message)
{
      call processMessages method

      if (is simulation mode)
            call emulateRealElementSynchronization method
      else
            call synchronizeWithRealElement method

      call processRegularActions method

      if (is simulation mode)
            call checkSimulationTimer method
}

call endLife method
```

.

Each agent basically iterates the following actions continuously until it receives a termination message:

- interpret and respond to the message it received (processMessage method);
- synchronize its state with a corresponding real-world element (synchronizeWithRealElement method);
- do predefined instructions depending on its current state (processRegularActions method).

Before terminating its life, an agent sends its processing information to other required agents (endLife method); for example, PDSP agents send their characteristics information to the PDSP disassembly/reassembly manager agent except for the reassembled products (refer to section V.2.1.1).

J.2.1. processMessage method

The processMessage method handles received messages as follows:

- dispatch the earliest arrived message;
- compare the dispatched message with comprehensible message set one by one; in other words, compare the type and objective (refer to table VIII.1) of the message with that of elements in the predefined processable message set;
- in case the dispatched message is understandable, in other words, if there exists a matching message type and objective, react as predefined instructions; for example, change its state or answer to the request.

Hence the main routine of the processMessage method is as follows:

```
dispatch earliest arrived message

if (is dispatched message equal to the element 1 in the processable message set)
      act as instruction 1

else if (is dispatched message equal to the element 2 in the processable message set)
      act as instruction 2

…

else
      do nothing
```

.

For example, when a resource receives an estimated waiting time (EWT) request message (refer to table VIII.1 and steps 1 – 3 in figure VIII.4) from a PDSP, it returns a message on the calculated EWT to the requesting PDSP as follows:

```
…
else if (is header of dispatched message is  “RQ/EWT” )
{
      get a requesting operation ID
      get a historical waiting time of the requested operation under the current dispatching rule
      calculate EWT
      send message on EWT
}
else if …
```

.

J.2.2. synchronizeWithRealElement method

The synchronizeWithRealElement method is empty, because the current version did not implement the features related to the real-world remanufacturing system control. Elements in the real-world are not synchronized during simulations, hence the emulateRealElementSynchronization method generates external signals from the corresponding real-world element by itself based on statistical information. All the agents' external signal self-generations are coordinated by the internal timer in the simulator. For example, a resource agent generates an operation processing end signal by itself when the elapsed time of operation processing meets the statistically generated operation processing time which is decided when it starts the operation.

The internal timer suspends its time pass until all the living agents raise their flags indicating that they have nothing to do more at current time. The checkSimulationTimer method deals with raising the nothing to do flag; agent's current state is compared with its predefined nothing to do state set, and it raises the flag if a matching element exists. When the nothing to do flags of all agents are up, the timer passes its time by one unit time and drops all the flags back, and then each agent can do their works.

J.2.3. processRegularActions method

The processRegularActions method follows required instructions depending on the agent's current state, hence its main routine is as follows:

```
switch (current state)
{
      case state 1:    act as instruction 1
                       break

      case state 2:    act as instruction 2
                       break

      …
}
```

.

For example, EWT information collection and operation/resource selection part in the processRegularActions method of a PDSP agent is as follows:

```
…

case PDSP_Created:
case Operation_Completed:
            get possible next operations
            get resources in charge of the operations
            distribute EWT request message to the resources
            change state to Waiting_EWT
            break;

case EWT_Collected:
            select operation and resource
            register operation to selected resource
            change state to Waiting_Registration_Confirmation
            break

…
```

A PDSP's processMessage method links the instructions between before and after EWT information collection; when the method identifies a received message as the requested EWT information from a resource, it saves the EWT information, and then changes its state to the EWT information is collected if all resources responded to its request.

All the other agents have similar structure of the processMessage methods; each state has instruction sets to do and they are correlated with state change by messages from other agents or synchronization signals from the corresponding real-world elements which is substituted with self generated signals during simulations.

Appendix K. References

Akyol D.E. and Bayhan G.M. (2007), A review on evolution of production scheduling with neural networks, Computers & Industrial Engineering, 53(1): 95-122.

Arzi Y. and Iaroslavitz L. (1999), Neural network-based adaptive production control system for a flexible manufacturing cell under a random environment, IIE Transactions, 31(3): 217-230.

Babiceanu R.F. and Chen F.F. (2006), Development and applications of holonic manufacturing systems: a survey, Journal of Intelligent Manufacturing, 17(1): 111-131.

Baker K.R. (1984), Sequencing rules and due-date assignments in a job shop, Management Science, 30(9): 1093-1104.

Barata J., Camarinha-Matos L., and Candido G. (2008), A multiagent-based control system applied to an educational shop floor, Robotics and Computer-Integrated Manufacturing, 24(5): 597-605.

Barnett A. and Kleitman D.J. (1978), Some optimization problems with bulk-service queues, Studies in Applied Mathematics, 58: 277–290.

Bourgeois F., de Meneses Y.L., Chollet S.K., and Jacot J. (2005), Defining assembly specifications from product functional requirements using inertial tolerancing in precision assembly, Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP), 266-72.

Brandimarte P., Rigodanza M., and Roero L. (2000), Conceptual modeling of an object-oriented scheduling architecture based on the shifting bottleneck procedure, IIE Transactions, 32(10): 921-929.

Chandru V., Lee C.Y., and Uzsoy R (1993), Minimizing total completion-time on a batch processing machine with job families, Operations Research Letters, 13(2): 61-65.

Cheeseman M.J., Swann P., Hesketh G.B., and Barnes, S. (2005), Adaptive manufacturing scheduling: a flexible and configurable agent-based prototype, Production Planning and Control, 16(5): 479-487.

Cho H. and Lee I. (1999), Integrated framework of IDEF modelling methods for structured design of shop floor control systems, International Journal of Computer Integrated Manufacturing, 12(2): 113-128.

CIMPACT (2000), SIMAS II, The Assembly Simulator, Available: http://www.cimpact.ch/.

Dawood N.N. (1996), A simulation model for eliciting scheduling knowledge: An application to the precast manufacturing process, Advanced in Engineering Software, 25(2-3): 215-223.

Deb R.K. and Serfozo R.F. (1973), Optimal control of batch service queues, Advances in Applied Probability, 5: 340–361.

Demello L.S.H. and A.C. Sanderson (1990), And/or graph representation of assembly plans, IEEE Transactions on Robotics and Automation, 6(2): 188-199.

El-Bouri A. and Shah P. (2006), A neural network for dispatching rule selection in a job shop, International Journal of Advanced Manufacturing Technology, 31(3-4): 342-349.

ElMekkawy T.Y. and ElMaraghy H.A. (2003), Real-time scheduling with deadlock avoidance in flexible manufacturing systems, International Journal of Advanced Manufacturing Technology, 22(3-4): 259-270.

Enrico (2008), Erico's maserati pages, Available: http://www.maserati-alfieri.co.uk/.

Fowler J.W., Phillips D.T., and Hogg G.L. (1992), Real-time control of multiproduct bulk-service semiconductor manufacturing processes, IEEE Transactions on Semiconductor Manufacturing, 5:158–163.

Franke C., Basdere B., Ciupek M. and Seliger S. (2006), Remanufacturing of mobile phones-capacity, program and facility adaptation planning, Omega, 34(6): 562-570.

Glassey C.R. and Weng W.W. (1991), Dynamic batching heuristic for simultaneous processing, IEEE Transactions on Semiconductor Manufacturing, 4: 77–82.

Guide V.D.R. (1996), Scheduling using drum-buffer-rope in a remanufacturing environment, International Journal of Production Research, 34(4): 1081-1091.

Guide V.D.R., Kraus M.E., and Srivastava R. (1997), Scheduling policies for remanufacturing, International Journal of Production Research, 48(2): 187-204.

Guide V.D.R., Souza G.C., and van der Laan E. (2005), Performance of static priority rules for shared facilities in a remanufacturing shop with disassembly and reassembly, European Journal of Operational Research, 164(2): 341-353.

Guide V.D.R., Srivastava R., and Kraus M.E. (1998), Proactive expediting policies for recoverable manufacturing, Journal of Operational Research Society., 49(5): 479-491.

Guide V.D.R., Srivastava R., and Kraus M.E. (2000), Priority scheduling policies for repair shops, International Journal of Production Research, 38(4): 929-950.

Haq A.N. and Ramanan T.R. (2006), A bicriterian flow shop scheduling using artificial neural network, International Journal of Advanced Manufacturing Technology, 30(11-12): 1132-1138.

Hirose M. (1990), Development of the holonic manipulator and its control, Proceedings of the IEEE Conference on Decision and Control, 1: 91–96.

Holthaus O. and Rajendran C. (1997), Efficient dispatching rules for scheduling in a job shop, International Journal of Production Economics, 48(1): 87-105.

Hsieh F.S. (2005), Automated negotiation based on contract net and Petri net, Lecture Notes in Computer Science, 3590: 148-157.

Hsieh F.S. (2008), Robustness analysis of holonic assembly/disassembly processes with Petri nets, Automatica, 44(10): 2538-2548.

Ijomah W.L. and Childe S.J. (2007), A model of the operations concerned in remanufacture, International Journal of Production Research, 45(24): 5857-5880.

Ikura Y. and Gimple M. (1986), Efficient scheduling algorithms for a single batch processing machine, Operations Research Letters, 5(2): 61-65.

Imamura S., Masaki H., and Nakazawa Y. (2001), Disassembly planning for machine maintenance with multi-agent cooperation, Proceedings of the IEEE International Symposium on Assembly and Task Planning, 214-219.

Ip W.H. (1997), Rule-based ARIMA models for FMS, Journal of Materials Processing Technology, 66(1-3): 240-243.

Jeng M., Xie S.L., and Peng M.Y. (2002), Process nets with resources for manufacturing modeling and their analysis, IEEE Transactions on Robotics and Automation, 18(6): 875-889.

Kadar B., Monostori L., and Szelke E. (1998), An object-oriented framework for developing distributed manufacturing architectures, Journal of Intelligent Manufacturing, 9(2): 173-179.

Kerr R.M. and Ebsary R.V. (1998), Implementation of an expert system for production scheduling, European Journal of Operational Research, 33(1): 17-29.

Kim K.H., Bae J.W., Song J.Y., and Lee H.Y. (1996), A distributed scheduling and shop floor control method, Computers & Industrial Engineering, 31(3-4): 583-586.

Kis T., Kiritsis D., Xirouchakis P., and Neuendorf K.P. (2000), A Petri net model for integrated process and job shop production planning, Journal of Intelligent Manufacturing, 11(2): 191-207.

Koestler A. (1970), Beyond atomism and holism, Perspectives in Biology and Medicine, 13(2): 131-&.

Kopacek P. and Kopacek B. (2006), Intelligent, flexible disassembly, International Journal of Advanced Manufacturing Technology, 30(5-6): 554-560.

Kreng V.B. and Tseng-Pin L. (2004), Modular product design with grouping genetic algorithm - a case study, Computers & Industrial Engineering, 46(4): 443-60.

Krill M. and Thurston D.L. (2005), Remanufacturing: Impacts of sacrificial cylinder liners, Journal of Manufacturing Science and Engineering, Transactions of the ASME, 127(3): 687-697.

Krothapalli N.K.C., and Deshmukh A.V. (1999), Design of negotiation protocols for multi-agent manufacturing systems, International Journal of Production Research, 37(7): 1601-1624.

Law A.M. and Kelton W.D. (1984), Confidence intervals for steady-state simulations. I. A survey of fixed sample size procedures, Operations Research, 32(6): 1221-1239.

Lee C.Y., Uzsoy R., and Martinvega L.A. (1992), Efficient algorithms for scheduling semiconductor burn-in operations, Operations Research, 40(4): 764-775.

Li Y., Chen J., and Cai X. (2007), Heuristic genetic algorithm for capacitated production planning problems with batch processing and remanufacturing, International Journal of Production Economics, 105(2): 301-317.

Lim M.K. and Zhang D.Z. (2004), An integrated agent-based approach for responsive control of manufacturing resources, Computers & Industrial Engineering, 46(2): 221-232.

Liu H.J. and Dong J.J. (1996), Dispatching rule selection using artificial neural networks for dynamic planning scheduling, Journal of Intelligent Manufacturing, 7(3): 243-250.

Macchiaroli R. and Riemma S. (2002), A negotiation scheme for autonomous agents in job shop scheduling, International Journal of Computer Integrated Manufacturing, 15(3): 222-232.

Marin C.A.G., Soto E.V., and Sanchez A.D. (2005), Describing an IMS by a FNRTPN definition: a VHDL approach, Robotics and Computer-Integrated Manufacturing, 21(3): 241-247.

Martinez M., Pham V.H., and Favrel, J. (1997), Dynamic generation of disassembly sequences, IEEE Symposium on Emerging Technologies & Factory Automation, ETFA, 177-182.

Mathirajan M. and Sivakumar A.I. (2006), A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, International Journal of Advanced Manufacturing Technology, 29(9-10): 990-1001.

McKay A., de Pemington A., and Baxter J. (2001), Requirements management: a representation scheme for product specifications, Computer Aided Design, 33(7): 511-20.

McPherson R.F. and White K.P. (2006), A framework for developing intelligent real-time scheduling systems, Human Factors and Ergonomics in Manufacturing, 16(4): 385-408.

Mertins K., Jochem R., and Jakel F.W. (1997), A tool for object-oriented modelling and analysis of business processes, Computers in Industry, 33(2-3): 345-356.

Naso D. and Turchiano B. (2004), A coordination strategy for distributed multi-agent manufacturing systems, International Journal of Production Research, 42(12): 2497-2520.

Neale J.J. and Duenyas I. (2000), Control of manufacturing networks which contain a batch processing machine, IIE TRANSACTIONS, 32(11): 1027-1041.

Neuts M.F (1967), A general class of bulk queues with Poisson input, Annals of Mathematical Statistics, 38(3): 759-770.

Noble J.S. and Lim H.H (2002), An evaluation of facility layout alternatives for a remanufacturing environment, Proceedings of the SPIE - The International Society for Optical Engineering, 4569: 158-66.

Odonoghue D., Healy E., and Sorensen H. (1994), Development of a rule-based finite-capacity scheduling system, Internaitional Journal of Production Economics, 36(2): 221-227.

Odrey N.G. and Mejia G. (2003), A re-configurable multi-agent system architecture for error recovery in production systems, Robotics and Computer-Integrated Manufacturing, 19(1-2): 35-43.

Ouelhadj D., Hanachi C., Bouzouia B., Moualek A., and Farhi A. (1999), A multi-contract net protocol for dynamic scheduling in flexible manufacturing systems (FMS), Proceedings 1999 IEEE International Conference on Robotics and Automation, 2: 1114-1119.

Pavliska A. and Srovnal V. (2002), Robot disassembly process using multi-agent system, From Theory to Practice in Multi-agent Systems (Lecture Notes in Artificial Intelligence), 2296: 227-233.

Pflughoeft K.A., Hutchinson G.K., and Nazareth D.L. (1996), Intelligent decision support for flexible manufacturing: Design and implementation of a knowledge-based simulator, Omega-International Journal of Management Science, 24(3): 347-360.

Poon C.K. and Zhang P.X. (2004), Minimizing makespan in batch machine scheduling, Algorighmica, 39(2): 155-174.

Ramasesh R. (1990), Dynamic job shop scheduling - a survey of simulation research, OMEGA-International Journal of Management Science, 18(1): 43-57.

Ramaswamy S. and Yi Y. (1999), Interactive modeling and simulation of virtual manufacturing assemblies: an agent-based approach, Journal of Intelligent Manufacturing, 10(6): 503-18.

Reaidy J., Massotte P., and Diep D. (2006), Comparison of negotiation protocols in dynamic agent-based manufacturing systems, International Journal of Production Economics, 99(1-2): 117-130.

Reyes A., Yu H., Kelleher G., and Lloyd S. (2002), Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS, Computers in Industry, 47(1): 123-138.

Roy D., Anciaux D., and Vernadat F. (2001), SYROCO: A novel multi-agent shop-floor control system, Journal of Intelligent Manufacturing, 12(3): 295-307.

Ryan J. and Heavey C. (2006), Process modeling for simulation, Computers in Industry, 57(5): 437-450.

Sakara D. (2006), Modeling and scheduling of remanufacturing systems, Ph.D. dissertation, STI-IPR-LICP, EPFL, Lausanne, Switzerland.

Sandholm, T.W. (2000), Automated contracting in distributed manufacturing among independent companies, Journal of Intelligent Manufacturing, 11(3): 271-83.

Shen C. and Zhang X. (2008), Supply planning model for remanufacturing system based on multi-agent technology, Proceedings of the IEEE International Conference on Automation and Logistics (ICAL), 357-361.

Shen W.M. and Norrie DH (2001), Dynamic manufacturing scheduling using both functional and resource related agents, Integrated Computer-aided Engineering, 8(1): 17-30.

Shen W.M., Hao Q., Wang S.Y., Lia Y.S., and Ghenniwa H (2007), An agent-based service-oriented integration architecture for collaborative intelligent manufacturing, Robotics and Computer-Integrated Manufacturing, 23(3): 315-325.

Shen W.M., Hao Q., Yoon H.J., and Norrie D.H. (2006), Applications of agent-based systems in intelligent manufacturing: An updated review, Advanced Engineering Informatics, 20(4): 415-431.

Shiue Y.R. (2009), Development of two-level decision tree-based real-time scheduling system under product mix variety environment, Robotics and Computer-Integrated Manufacturing, 25(4-5): 709-720.

Shnits B. and Sinreich D. (2006), Controlling flexible manufacturing systems based on a dynamic selection of the appropriate operational criteria and scheduling policy, International Journal of Flexible Manufacturing Systems, 18(1): 1-27.

Smith R.G. (1980), The contract net protocol - high-level communication and control in a distributed problem solver, IEEE Transactions on Computers, 29(12): 1104-1113.

Song C., Guan X.H., Zhao Q.C., and Ho Y.C. (2005), Machine learning approach for determining feasible plans of a remanufacturing system, IEEE Transactions on Automation Science and Engineering, 2(3): 262-275.

Sousa P. and Ramos C. (1999), A distributed architecture and negotiation protocol for scheduling in manufacturing systems, Computers in Industry, 38(2): 103-113.

Stanfield P.M., King R.E., and Hodgson T.J. (2006), Determining sequence and ready times in a remanufacturing system, IIE Transactions, 38(7): 597-607.

Steinhilper R. (1998), Remanufacturing: the ultimate form of recycling, Stuttgart Germany: Fraunhofer IRB Verlag.

Sung C.S., Choung Y.I., Hong J.M., and Kim Y.H. (2002), Minimizing makespan on a single burn-in oven with job families and dynamic job arrivals, Computers & Operations Research, 29(8): 995-1007.

Tateno T. and Kondoh S. (2005), Proposal of ubiquitous disassembly system for realizing reuse and recycling in cooperative distributed facilities, Proceedings - Fourth International Symposium on Environmentally Conscious Design and Inverse Manufacturing, Eco Design 2005, 208-209.

Trappey A.J.C., Lin G.Y.P., and Ku C.C. (2007), Design and analysis of a rule-based knowledge system supporting intelligent dispatching, International Journal of Advanced Manufacturing Technology, 35(3-4): 385-93.

Valckenaers P., Sauter J., Sierra C., and Rodriguez-Aguilar J.A. (2007), Applications and environments for multi-agent systems, Autonomous Agents and Multi-agent systems, 14(1): 61-85.

Van der Zee D.J. (2002), Adaptive scheduling of batch servers in flow shops, International Journal of Production Research, 40(12): 2811-2833.

Van der Zee D.J. (2004), Dynamic scheduling of batch servers with compatible product families, International Journal of Production Research, 42(22): 4803-4826.

Veerakamolmal P. and Gupta S.M. (2002), A case-based reasoning approach for automating disassembly process planning, Journal of Intelligent Manufacturing, 13(1): 47-60.

Voutsinas T.G. and Pappis C.P. (2002), Scheduling jobs with values exponentially deteriorating over time, International Journal of Production Economics, 79(3): 163-169.

Wang D.S., Nagalingam S.V., and Lin G.C.I. (2007), Development of an agent-based Virtual CIM architecture for small to medium manufacturers, Robotics and Computer-Integrated Manufacturing, 23(1): 1-16.

Wong T.N., Leung C.W., and Tang H.P. (2008), A multi-agent system framework for manufacturing planning and control, 7th World Congress on Intelligent Control and Automation, 410-415.

Wong T.N., Leung C.W., Mak K.L., and Fung R.Y.K. (2006), Dynamic shopfloor scheduling in multi-agent manufacturing systems, Expert Systems with Applications, 31(3): 486-494.

Xie X.L. and Jeng M.D. (1999), ERCN-merged nets and their analysis using siphons, IEEE Transactions on Robotics and Automation, 15(4): 692-703.

Xue D., Sun J., and Norrie D.H. (2001), An intelligent optimal production scheduling approach using constraint-based search and agent-based collaboration, Computers in Industry, 46(2): 209-231.

Yildirim M.B., Cakar T., Doguc U., and Meza JC (2006), Machine number, priority rule, and due date determination in flexible manufacturing systems using artificial neural networks, Computers & Industrial Engineering, 50(1-2): 185-194.

Zhang D.Z., Anosike A., and Lim M.K. (2007), Dynamically integrated manufacturing systems (DIMS) - a multiagent approach, IEEE Transactions on Systems, Man, and Cybernetics - Part A (Systems & Humans), 37(5): 824-850.

Zhang Y.X. and Chen H. (1999), A knowledge-based dynamic job-scheduling in low-volume/high-variety manufacturing, Artificial Intelligence in Engineering, 13(3): 241-249.

Zhang Z., Cheng T., Wu B., and Yang S. (2003), Price-based negotiation for task assignment in a distributed network manufacturing mode environment, International Journal of Advanced Manufacturing Technology, 21(2): 145-156.

Zussman E. and Zhou M.C. (1999), A methodology for modeling and adaptive planning of disassembly processes, IEEE Transactions on Robotics and Automation, 15(1): 190-194.

Zwingmann X., Ait-Kadi D., Coulibaly A., and Mutel B. (2008), Optimal disassembly sequencing strategy using constraint programming approach, Journal of Quality in Maintenance Engineering, 14(1): 46-58.

Curriculum Vitae

Education

2005.09 – Present
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
Ph.D. candidate in Manufacturing systems and robotics (CAD and Production)
- Thesis: *Quality Embedded Intelligent Remanufacturing*

1998.03 – 2000.02
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
Master of Engineering in Industrial Engineering (Concurrent Engineering)
- Thesis: *PPO (Product-Process-Organization) Reference Model for the Product Development Information Management*

1994.03 – 1998.02
KAIST, Daejeon, Korea
Bachelor of Science in Computer Science

Work experience

2001.08 – 2005.08
F1 Consulting (LKFS), Seoul, Korea
Senior consultant, retail risk management division
Co-researcher, credit risk management system development center

2000.01 – 2001.07
Computronics (Synergy consulting), Seoul, Korea
Team leader, financial engineering division

Publications

Journal

Published
- Hong-Bae Jun, Jong-Ho Shin, *Youngseok Kim*, Dimitris Kiritsis, and Paul Xirouchakis, A framework for RFID applications in product lifecycle management, International Journal of Computer Integrated Manufacturing, 22(8), 595-615, 2009

Submitted
- Youngseok Kim, Dimitris Kiritsis, and Paul Xirouchakis, Quality embedded dispatching rules for remanufacturing systems, International Journal of Production Research
- Youngseok Kim, Dimitris Kiritsis, and Paul Xirouchakis, Knowledge-based dynamic priority dispatching rule allocation for real-time repair/remanufacturing system scheduling: a heuristic for knowledge acquisition, International Journal of Production Research
- Youngseok Kim, Dimitris Kiritsis, and Paul Xirouchakis, Information framework for quality embedded remanufacturing system, International Journal of Production Planning and Control

Conference

International conference
- *Youngseok Kim*, Hong-Bae Jun, Dimitris Kiritsis, and Paul Xirouchakis, Knowledge-based Multi-agent Framework for Products End-of-life (EOL) Scheduling, Advances in Production Management Systems Conference of the IFIP Working Group 5.7, Espoo, Finland, 2008
- *Youngseok Kim*, Hong-Bae Jun, Dimitris Kiritsis, and Paul Xirouchakis, A methodology for modeling quality embedded remanufacturing system, Advances in Production Management Systems Conference of the IFIP Working Group 5.7, Linköping, Sweden, 2007
- Hyo-Won Suh, Heejung Lee, *Youngseok Kim*, Hong-Bae Jun, and Seungchul Ha, An object-oriented representation for PPO information, Proceedings of the 2nd Asia-Pacific Industrial Engineering and Management System (APIEMS) Conference, Kanazawa, Japan, 1999
- *Youngseok Kim* and Hyo-won Suh, An integrated methodology of BPR-ISP-A/D based on PPO object evolution, CALS/EC KOREA`99 Proceedings of International Conference, Seoul, Korea, 1999

Korean local conference

- Hyo-won Suh, Hong-bae Jun, and *Youngseok Kim*, A hierarchical process schema for integrated management of project and workflow, Proceedings of Korean Society of CAD/CAM Engineers, 2000
- Hyo-won Suh and *Youngseok Kim*, An integrated PPO reference model for PDM kernel, Proceedings of Korean Society of CAD/CAM Engineers, 2000
- Hyo-won Suh, Hong-Bae Jun and *Youngseok Kim*, Layered process schema for integrated management of the process and workflow, Proceedings of Korean Society of CAD/CAM Engineers, 2000
- Hyo-won Suh and *Youngseok Kim*, An approach of layered virtual structure (LVS) for dynamic product information, Proceedings of Korean Society of CAD/CAM Engineers, 2000
- Hyo-won Suh and *Youngseok Kim*, A study on UML application for PDM construction and development, Proceedings of Korean Society of CAD/CAM Engineers, 1999
- Hyo-won Suh, Seungchul Ha, Heejung Lee, and *Youngseok Kim*, A study on workflow modeling for PDM construction and development, Proceedings of Korean Society of CAD/CAM Engineers, 1999
- Hyo-won Suh and *Youngseok Kim*, A study on UML application to motor development process, Proceedings of Korean Institute of Industrial Engineers, 1998

Project participation

Business area

2004.08 – 2005.05 **Advanced credit risk management strategy & system development, Hyundai card**
Consultant

2003.05 – 2004.03 **Retail loan early warning system development, Korea exchange bank**
Consultant (project leader)

2003.04 – 2003.10 **Asset loss distribution model construction, Korea export insurance corporation**
Assistant (off-site, assistant in model parameter estimation)

2002.12 – 2003.04 **Total asset management system development, SK life insurance**
Consultant

2002.07 – 2003.01 **Credit risk management strategy & system design, Samsung life insurance**
Consultant (project leader), System designer

2002.05 **Retail loan data management support, Hana bank**
System engineer

2002.02 – 2002.04 **Market risk management system upgrade, Hanmi bank**
System customizer

2000.06 – 2000.11 **Stock portal system development, Maekyung internet**
System designer & developer (real-time stock market data management)

Academic area

2005.09 – 2008.06 **Product lifecycle management and information tracking using smart embedded systems (PROMISE), IMS 01008 / EU FP6-IP-507100, EU consortium**
Researcher

1998.12 – 1999.12 **Research on concurrent product development process, Hyosung electronics**
Researcher, System modeler/developer

1998.06 – 1999.10 **Information structure design for the machine design and evaluation, Tongil heavy industries**
Researcher