An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the
8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM’95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the
8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM’95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the 8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM’95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the
8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe '94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe '94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM'95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the
8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM’95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the 8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe ’94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM’95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989

An Enhanced Scenario Modelling Technique for Object-Oriented Analysis

Position Paper

Submitted to, accepted by but not presented at the
8th International Workshop on Software Specification and Design

Dorothea Beringer

Swiss Federal Institute of Technology
Software Engineering Laboratory
EPFL-DI-LGL
CH - 1015 Lausanne

Abstract

During the last years, scenarios, use-cases and other techniques for modelling the global behaviour have been introduced into object-oriented analysis. However, aggregation and generalisation of scenarios as well as the integration of scenarios reflecting different abstraction levels are usually not supported. By integrating these concepts into an enhanced scenario modelling technique we could overcome some of the drawbacks of current approaches, e.g. their limitation to monolithic systems with fixed automation boundaries, their restriction to flat event structures, and their insufficient support of the transition from analysis to design.

Introduction

The need for modelling the global behaviour is not at all new or unique to object oriented analysis. Already in structured analysis, the event-oriented decomposition [McMenamin84] and the modelling of business functions have been well accepted in industrial projects, in spite of certain weaknesses in the approaches. Yet in the first object-oriented methods the emphasis lied mainly on modelling the local behaviour. The approaches started directly with finding and specifying the problem domain objects of the final software system. Scenarios (e.g. as event trace diagrams [Rumbaugh91]) were mentioned only as an additional help for making the other models of analysis. Certain concepts like using an environmental model similar to the one propagated by modern structured analysis [Yourdon89], or like using controller objects to encapsulate overall behaviour, were refused by many methodologists as being functional and not object-oriented, although others regarded them as indispensable [Mueller93, Berard93].

Many developers in industrial projects, however, missed a starting point for finding objects and a way to bridge the gap between a vague problem statement and the specification of a system of interacting objects. They kept asking for models that show the global behaviour, because with an analysis model that only shows the local behaviour of individual objects, they could neither define the functional requirements in a way understandable and validatable by users, nor trace these requirements to the specification of individual objects [Beringer93, Mueller93].

As a consequence, the techniques for modelling the global behaviour as introduced by OOSE (use-cases and object interaction diagrams [Jacobson92]) or by Fusion (system operations, life-cycle expressions and object interaction graphs [Coleman94]) have been greatly accepted by industry

and have also influenced the new versions of Booch [Booch94] and OMT [Rumbaugh94]. Other techniques that use scenarios to start the modelling process and to find the responsibilities of objects are e.g. found in [Rubin92, Graham94, Waldén95].

Some drawbacks of current approaches

Many of the current approaches to modelling global behaviour in object-oriented analysis have common weaknesses. These drawbacks do not make it impossible to use the methods, but depending on the specific circumstances of a project, they can cause frustration, delays and additional costs. In the following, some of them are listed:

- **Flat event structure:** It is assumed that the external events, which trigger the use-cases, system operations or scenarios, do not need any further structuring. Dependencies and hierarchical relationships between events are of little or no interest, and modularisation is not supported at all.
- **Only one level of abstraction:** Most techniques support only one abstraction level concerning scenarios and external events. This level must be determined right away at the beginning of the modelling process. Under certain circumstances, however, it would be convenient to represent several abstraction levels in one model. Also this would allow to support an iterative process for modelling and integrating different user's views, for gradually advancing from an informal and opaque model to a more formal and complete model, and for finding the best suited abstraction level for the transition to the design (see also the challenges that requirements engineering will have to meet in the future as described in [Jarke94]).
- **Fixed automation boundaries:** Often an analysis model of the problem domain (also called essential model or technology free model) needs to be made before the automation boundaries of the software solution are known. Also it may happen that the automation boundaries shift during the process of requirements determination. In both cases it should be possible to have a model of the global behaviour that can be adapted easily.
- **Matrix-approach:** The scenario specification is often complemented by an object-model that focuses only on data. Thus the scenarios and the data-entities form a matrix, similar to a CRUD-matrix in structured analysis, an object-event-table in M.E.R.o.DE [Dedene94], or the read- and change-clauses in Fusion [Coleman94]. Whether such an approach can still be called object-oriented may be doubted.

  Moreover current approaches are focused on monolithic systems. Only two levels of granularity are considered: the external behaviour of the system as a whole, and the behaviour of the individual objects on the level of object-oriented programming languages. Global behaviour that is composed of the global behaviour of several subsystems is not handled. Also the interactions between these subsystems are not modeled. More appropriate were a module-approach which would support several levels, and which could provide the same granularity for both the static and the dynamic views by integrating them into modules.
- **Bias towards data modelling and functional decomposition:** Though starting with an analysis model that focuses on the global behaviour, some of the development processes

quite often produce design object models of minor quality. Such designs tend to be biased towards data modelling [Sharble93] and/or functional decomposition [Waldén95].

- **Unclear transition to design:** With a solution-oriented analysis model and external events and scenarios that are modelled on a technical level, the transition to the design is straightforward (see e.g. [Coleman94]). However if the analysis has been done without any knowledge about the user interface of the future software system, or if the interaction mechanism or even automation boundaries have been abstracted away, then a higher conceptual level might be represented in the analysis model. In this case, no adequate techniques and guidelines are provided for proceeding to the model of the actual interface mechanism. Also no support is given for showing the relations between the analysis and the design models.

Some of the above obstacles can be addressed by a better awareness of the effective limits of seamlessness [Beringer94], by a better understanding of the goals of the different models, and by a project specific adaptation of methods and modelling techniques [Beringer95]. But in addition, enhanced scenario modelling techniques are needed. These should allow the modelling of global behaviour on different abstraction levels by using modular and hierarchical scenarios, and also provide a well defined transition between these levels.

Characteristics of an enhanced scenario modelling technique

In one of our ongoing research projects we are working on the definition of an enhanced scenario modelling technique and on the integration of such a technique into current object-oriented analysis methods. This approach should overcome some of the mentioned drawbacks of current techniques for modelling global behaviour.

Important characteristics of the enhanced scenario modelling technique are:

- **Sequence of interactions:** A single scenario shows one possible sequence of interactions (or events) between several objects, between several subsystems (or modules), or between a system and its environment (agents).
- **One single concept for scenarios with and without intermediate interactions**: No guidelines are given concerning the starting and the termination criteria of one scenario. Each scenario is triggered by its stimuli, most often an external event. But one single scenario may well contain intermediate interactions between the system and its environment. The interactions are not restricted to only one stimuli and some output-events as is the case e.g. for the system operations in Fusion. This allows to have a single scenario concept that encompasses most of the differing concepts found in current approaches, e.g. scripts, use-cases, system operations and business functions. Scenario aggregation is used to map these concepts onto each other.
- **Scenario types**: Scenarios can be specified by scenario types. Such a specification states at least which events are involved and under which conditions these can occur. Additional information concerns possible event parameters, essential state changes and the permissible order of events. The instance of a scenario type is determined by the parameter values of the external events and by the system state.
- **Scenario aggregation**: A sequence of scenarios over the same system may be aggregated into one single scenario (aggregation in time). If several scenarios which are defined over different components or subsystems are combined into one single scenario over the whole system, we have an aggregation in space. Of course, certain conditions concerning com-

pleteness and consistency of the events involved need to be satisfied. The integration of the aggregation of subsystems and of the aggregation of their corresponding scenarios leads to a composition of behaviour similar to the one described in [Adams94].

Any scenario aggregation has to take into account and has to model the possible order of its component scenarios: sequence, alternation, concurrency and iteration.

- **Scenario generalisation**: Event types as well as scenario types may be put into a generalisation hierarchy. Generalisation allows to abstract away details like alternative courses or additional event-parameters. Generalisation may also be used to show the relationships between scenarios on different abstraction levels (e.g. between those scenarios using only conceptual events and those scenarios on a technical level that correspond to user-interactions in the software-solution).

Scenario aggregation and generalisation not only allow to model systems with a complex structure of external events, but also to start the modelling process at any abstraction level that seems convenient, and to evolve the model towards that scenario specification which will be the basis for the design. More abstract views may be deduced at any time and are clearly related to the more detailed model. Changes in the definition of the system boundaries can be dealt with and documented by aggregation. Further, a well defined transition is provided between analysis models that reflect different views of the system, e.g. between views with conceptual and views with technical events, views with or without system boundaries, or internal and external system views.

A notation for representing scenario types is given by interaction diagrams (also called event-trace diagrams). The interaction diagrams must be decomposable and need to be enhanced by the possibility to show alternative interaction sequences and essential state-changes. Such a notation could be applied for analysis as well as design models, and for the external as well as the internal system views. Our goal is not to give a rigorous formal approach, instead we want to provide a semi-formal modelling technique that is adequate for the majority of developers and users that need to describe global behaviour. A textual notation as an alternative to the graphical one will be important, too.

Conclusions

Use-cases and scenarios have become very popular to start the process of object-oriented analysis. Yet most techniques neither support the aggregation or generalisation of scenarios, nor the modelling of several abstraction levels. Having these concepts in an enhanced scenario modelling technique would allow to overcome some of the drawbacks of current techniques for modelling global behaviour.

References

[Adams94] G. Adams, J.P. Corriveau; Capturing Object Interactions; Proceedings of TOOLS Europe '94, Versailles, 1994

[Berard93] E.V. Berard; Essays on Object-Oriented Software Engineering, Volume1; Prentice Hall, 1993

[Beringer93] D. Beringer; Der Weg zum Objektmodell; Output, Sonderausgabe über Objektorientierte Systeme, Nov. 1993

[Beringer94] D. Beringer; Limits of Seamlessness in Object-Oriented Software Development; Proceedings of TOOLS Europe '94, Versailles, 1994

[Beringer95] D. Beringer; The Model Architecture Frame: Quality Management in a Multi Method Environment; Proceedings of SQM'95, Seville, 1995

[Booch94] G. Booch; Object Oriented Analysis and Design; Benjamin/Cummings, 1994

[Coleman94] D. Coleman et al.; Object-Oriented Development: the Fusion Method; Prentice-Hall, 1994

[Dedene94] G. Dedene, M. Snoeck; M.E.R.o.DE: A Model-driven Entity-Relationship object-oriented DEvelopment method; ACM SIGSOFT, vol 19 no 3, 1994

[Graham94] I. Graham; Beyond the Use Case: Combining task analysis and scripts in object-oriented requirements capture and business process re-engineering; Proceedings of TOOLS Europe' 94, Versailles, 1994

[Jacobson92] I. Jacobson; Object-Oriented Software Engineering; Addison Wesley, 1992

[Jarke94] M. Jarke, K. Pohl; Requirements engineering in 2001: (virtually) managing a changing reality; Software Engineering Journal, Nov. 1994

[McMenamin84] M. McMenamin, J.F. Palmer; Essential Systems Analysis; Yourdan Press, 1984

[Mueller93] G. Mueller-Luschnat et al.; Objektorientierte Analyse und Geschäftsvorfallmodellierung; Proceedings of the EMISA-Fachtagung über objektorientierte Methoden für Informationssysteme, Universität Klagenfurt, 1993

[Rubin92] K.S. Rubin, A. Goldberg; Object Behavior Analysis; Communications of the ACM, Sept. 1992

[Rumbaugh91] J. Rumbaugh et al.; Object Oriented Modelling and Design; Prentice-Hall, 1991

[Rumbaugh94] J. Rumbaugh; Modeling & Design; JOOP, issues of June 1994, Sept 1994, Nov/Dec 1994, Febr 1995 and March/April 1995

[Sharble93] R. Sharble and S. Cohen; The Object-Oriented Brewery: A Comparison of Two Object-Oriented Development Methods; ACM SIGSOFT, April 1993

[Waldén95] K. Waldén, J.M. Nerson; Seamless Object-Oriented Software Architecture; Prentice Hall, 1995

[Yourdon89] E. Yourdon; Modern Structured Analysis; Prentice-Hall, 1989