Goals, Interpretations, and Policies in Information Systems Design

Gil Regev [1], Alain Wegmann [1]

1 Institute for computer Communication and Application (ICA)
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland
http://icawww.epfl.ch
{gil.regev, alain.wegmann}@epfl.ch

Abstract: Current goal-oriented requirements engineering methods focus on the definition of optimal requirements that an information system needs to support in order to help its stakeholders to achieve their goals. But, the lack of systemic reasoning and disregard for questions of interpretation lead to insufficient attention given to activities and implicit policies affecting the definition of these goals. This results in the optimization of the requirements for potentially inadequate goals. Our framework relates stakeholders' goals to the their activities, their policies and their interpreted constraints and capabilities. It enables requirements engineers to better understand the rationale for goals found through requirements elicitation techniques and shows that conflicting goals can be reconciled by understanding how they fit in a higher-level activity. This results in the formulation of a more adequate set of goals that the information system should support in order for the organization and stakeholders to perform their activities.

Index Terms - Goal-Oriented requirements engineering, information systems, complex systems, system science, modeling, implicit policies, interpretations, activities.

1 Introduction

The explanation of behavior in terms of goals, ends, purposes etc, dates back to at least Aristotle with the concept of Teleology. Teleology is not a favored term amongst scientists because of its use by vitalists in explaining the behavior of natural systems. Medawar and Medawar quoted by Checkland [9, p. 75] propose the alternative concept of Teleonomy, which is defined as, "*Behavior described as if it fulfilled a purpose is teleonomic*." Teleonomy is an important concept because it shows that purpose is a matter of interpretation on the part of the observer. Purpose is so characteristic of our behavior that we use it to try to understand phenomena that we find impossible to predict, which we call complex systems [5], [21], [48]. Information systems are built for human organizations, which can be thought of complex systems [8]. Human organizations are made of groups of people working together to achieve some common purpose. These groups of people have potentially conflicting purposes even though, most of the time, they also attempt to work toward a common purpose. We can model organizations as communities, which attempt to take purposeful action in an environment that they interpret is placing passive and active constraints on them [8]. These communities are made of members, also called sub-communities, which also attempt to take purposeful action within their environment.

Traditional approaches to requirements engineering have focused on the behavior that the system to be built had to exhibit in its interactions with the users. Over the last 20 years several goal-oriented approaches have been developed in the fields of requirements engineering [2], [14], [29], [38] software engineering [10], [17] and usability [12], [13], [22], [34]. Goal-oriented requirements engineering approaches promise to specify systems that better support stakeholders by analyzing their goals. Goals are viewed as the source of all requirements. Goals are considered to be, "*more stable than the requirements to achieve them*." [45] By focusing on goals rather than on interactions, it is also easier to investigate different ways of achieving the stated goals and to detect and solve conflicts between them.

In this paper we argue that in the field of enterprise information systems, the understanding of the notions of community, activity, perception, interpretation and implicit policies are crucial to defining the correct set of goals for the system to be built. Requirements engineering methods make the assumption that a system needs to be built for the problem to be solved. But, we believe that goal definition should be

approached with the aim of improving the organization rather than specifying the requirements for an information system. As noted by Markus and Keil [28], "*The goal of system building should be organizational improvement.*" Our approach aims at providing the necessary concepts for defining the goals for a system to be built but beginning without any assumptions about the existence or even the desirability of such a system. More specifically, our approach models observed reality in the following way:

1. Organizations can be modeled as networks of purposeful entities, which we call communities. These communities influence each other through activities that they perform together.
2. In the context of each of these communities, observed goals are not a given and they are not immutable. Most of the observable goals, even those called high-level in the literature, are the manifestation of policies that a given community has put in place over time and that can only be understood historically through an analysis of the role of the community and its interpretation its relationships with communities with which it performs its activities.
3. Members of a community generally subscribe to community goals but they also have private goals that, many times, come from other activities they perform with other communities. These private goals may conflict with the community goals when an information system is imposed taking into account only the latter.

Section 2 describes the related work and the problem we address. Section 3 describes the framework we propose to approach the problem. In Section 4 we discuss the implications of our framework and its rationale. Section 5 will illustrate our framework with an example; the CompuSys case described in detail in Markus and Keil's paper "*If we Build it They Will Come: Designing Information Systems That People Want to Use*" [28].

2 The Problem

Enterprise information systems are generally considered to be built in order to help organizations achieve their goals. Requirements Engineering (RE) is the discipline that seeks to produce a detailed description of an information system to be built so that it matches the goals of the organization for which it was built. As defined by van Lamsweerde & Letier [45], RE is, "*concerned with the elicitation of high-level goals to be achieved by the envisioned system, specifications of software behavior, and to their evolution.*"

In the RE community, goals are usually defined as objectives to be achieved by the system and its environment [14], [43], [44], [45] or a state of affairs that is deemed desirable by stakeholders [19], [38]. Some methods distinguish between kinds of goals, such as, private and system goals [14], personal, corporate, practical, and false goals [13], business and personal goals [32]. Goal refinement is the process through which goals are broken down into sub-goals. Goal abstraction proceeds in the opposite direction providing super-goals. Goals can be refined into sub-goals by asking how these goals should be achieved while super goals are found by asking why a certain goal is sought [44].

The KAOS approach [14], [43], [44], [45] consists of a formal framework based on temporal logic and AI refinement techniques where all terms such as goal and state are consistently and rigorously defined. The main emphasis of KAOS is on the formal proof that the requirements match the goals that were defined for the envisioned system. The Non-Functional Requirements (NFR) approach [29], [30] is based on the notion of softgoals rather than (hard) goals. A softgoal is satisficed rather than achieved [29]. Goal satisficing is based on the notion that goals are never totally achieved or not achieved. Therefore Mylopoulos et al [29] state that, "*softgoals are satisficed when there is sufficient positive and little negative evidence for this claim, and that they are unsatisficeable when there is sufficient negative evidence and little positive support for their satisficeability.*" Goal-Based Requirements Analysis Method (GBRAM) [2] and ESPRIT CREWS [38] are less formal than KAOS but focus more on goal definition and the linking of goals to stakeholders' actual needs. Pohl [32] and Kaindl [19] also address this seemingly difficult phase of defining initial goals for the system to support.

The above methods do not address the notions of communities, the activities they perform, their interpretation of the world and their implicit and explicit policies. Sutcliffe and Maiden [42] link goals with policies, stating that "*Lower level goals support achievement of higher level policy.*" Anton et al. [4] also

note that policies may be the source of goals. However, both approaches only deal with the issues of explicit policies. They do not consider policies within the context of communities and activities.

A field that does address some of the issues surrounding communities and activities is enterprise modeling [23], [24], [36], [40], [41]. Enterprise modeling is interested in the definition of distributed information systems for organizations. It thus defines the concepts of community, activity, role, policy etc. A policy is usually defined as a set of rules related to a particular purpose or situation [23], [24], [25], [35], [36], [41]. The rules are described in terms of Deontic Logic in order to enable formal reasoning. Unfortunately, this view doesn't take into account that most policies are implicitly shared within a community and that perceptions and interpretations within a community usually lead to the definition of these policies.

We have found three problems that affect the goal-oriented RE field and a fourth problem that is applicable to requirements engineering in general. The three first problems are neatly summarized by Rolland et al. [38] who report on the difficulties of implementing goal-oriented methods in practice, "*practical experience...shows that goals are not given and therefore the question as to where they originate from...acquires importance. In addition, enterprise goals, which initiate the goal process do not reflect the actual situation but an idealized environmental one. Therefore, proceeding from this may lead to ineffective requirements. Thus, goal discovery is rarely an easy task. Additionally, it has been shown…that the application of goal reduction methods…to discover the component goals of a goal, is not as straightforward as the literature suggests.*" The fourth problem is due to the nature of RE, which is the construction of requirements for the purpose of building an information system.

Problem 1, goal discovery is not a simple task: The search for the "right" set of goals for which the system should be designed is indeed a complex task. Goal-oriented RE methods do not give enough importance to the origin of the goals they elicit (i.e. extract). Goals are usually elicited from scenarios, interview transcripts, formal policy statements, corporate goals, mission statements etc. [2]. Indeed, few attempts are made to analyze where these goals come from and to verify that these goals are indeed the goals for which the information system should be designed. In other words, it is taken for granted that goals elicited from the above sources (or a subset of these goals) are the goals the information system should support. Unfortunately, by not challenging the elicited goals, one runs the risk of building a well-tuned information system for the wrong goals.

Problem 2, expressed corporate goals do not provide a good starting point for RE: The main point here is that corporate goals describe an idealized setting, where only goals of the organization as a whole are considered. Basing the requirements for an information system on these goals - without addressing the constraints and private goals affecting communities of people who will use the system - leads to the specification of systems that fail to adequately support these communities and hence the organization as a whole.

Problem 3, goal reduction is not straightforward: Where enterprise information systems are concerned, goal reduction or goal refinement cannot be conceived as a mechanistic process. Rather, goal refinement is driven by policies, which are often implicit and are different for each organization. Not taking these implicit policies into account when specifying the system to be built will likely produce unacceptable solutions.

Problem 4, requirements engineering begins with the assumption that a system is needed: Requirements engineering, whether goal-oriented or not, generally places the system to be built at the center of the stakeholders' preoccupations. As can be seen from Rolland et al.'s [38] description of the contextual level, the phase in which requirements engineering begins, "*At the contextual level, it is of major importance to explore as many design alternatives as possible i.e., to visualize the various alternative ways by which a system can help an organization to achieve one of its objectives*." Specifying that an information system is needed is already a design choice, which is unnecessary and potentially harmful at the early stages of requirements engineering. Indeed this conditioning of stakeholders (including users and engineers) prevents the stakeholders from exploring solutions, which may be more adequate for the organization but which does not include a system to be built.

The solutions found in the goal-oriented RE literature propose to overcome these problems by coupling goal-oriented methods with scenarios [2], [19], [32], [38]. These approaches do help to find and justify goals, but these goals may still be inadequate because scenarios are often based on implicit purposes and implicit policies. Thus, the use of scenarios is not designed to identify higher-level goals, which would be better for the system to support. For example, the typical higher-level goal of scheduling a meeting is to solve a problem or make a collective decision. This higher-level goal will probably not be mentioned up in scenarios where people are asked to describe how they would go about scheduling a meeting. Designing a meeting support system with the high-level goal of scheduling meetings is pursuing the wrong goal as described by Cooper [13]. The problem of implicit policies is that scenarios embody organizational policies that remain largely implicit, or when they are made explicit, their rationale remains implicit. Leaving the rationale for goals and policies implicit means that we are not able to challenge them. Furthermore, without the rationale we are tempted to discard the policies and goals we are not comfortable with and keep those we are comfortable with. Also, scenarios often describe the interaction of users with the existing or future information system and hence cannot represent the bigger picture needed for organizational improvement.

Our framework addresses the above problems by proposing that requirements engineers focus their attention on policies (especially implicit policies) that are responsible for goals and their refinements (problem 1 and 3). Problem 2 is addressed by understanding that corporate goals cannot be refined into requirements by a mechanistic process because the different communities which carry out these goals have their own interpretations of what these goals mean and how they should be carried out. These interpretations form the basis of their policies. Within an organization, interpretations and policies often conflict (for example the conflicts that we take as granted between marketing, engineering, sales and manufacturing). We propose to reconcile these conflicts by addressing the needs of the overall activity performed by the different communities while not forgetting their private interpretations and policies. By not focusing on the building of the information system, we address problem 4.

The next Section describes our proposed framework. For purposes of clarity and brevity, this description contains only minimal background information and references. For a detailed explanation of the origins and implications of the framework please refer to the Discussion Section.

3 Framework Definition

Our framework integrates concepts that are studied in several disciplines such as, system science, software engineering, goal-oriented requirements engineering and social science. The aim of the framework is to define the relationships between a minimal set of concepts, thus enabling an informed discussion among stakeholders about their goals. This discussion eventually leads to the definition of goals that an information system could help them achieve. These goals can then serve for the definition of the requirements of the information system. At least initially, we do not assume that an information system is necessary. Hence, rather than ask how an information system can help the stakeholders to achieve their goals, we ask what the problem is. We then proceed by asking questions about activities the stakeholders perform, their goals and policies, their constraints and capabilities. In the course of this discussion, pre-conceived goals will likely be challenged leading to the specification of alternative goals to be pursued.

Organizations can be modeled as communities that represent complex systems [8], [21]. These communities can be represented at multiple organizational levels. An organization level represents a level of perception of the observer. At a given level the observer can model communities and activities as atomic, i.e. a concept in which sub-parts are not described, also called a whole. At a lower level these communities and activities can be modeled as compositions of sub-communities and/or sub-activities. As an illustration consider Figure 1 and Figure 2 where “Community 2” performing “Activity 2” is considered as atomic while in Figure 2 both the activity and the community are considered as composites. The result is that we see the sub-communities, “Community 3” and “Community 4”, of which “Community 2” is composed. We also see the activity performed by “Community 2”, that is “Activity 2” as a composite. This enables us to model the roles, or sub-activities performed by “Community 3” and “Community 4.” We use the concept of policy to define the sub-communities and the sub-activities. A policy is a plan that describes the sub-activities that need to be performed within an activity of interest, as well as which sub-community, if any, performs which sub-activity. These sub-activities represent the roles of these communities with the overall activity.

In our figures we use a notation derived from UML [31]. In this notation rectangles represent entities or concepts, ellipses represent activities, rectangles surrounding concepts and activities represent communities. For reasons of clarity we choose to represent only some of the concepts in a given community. Thus, all our models should be understood as partial models.

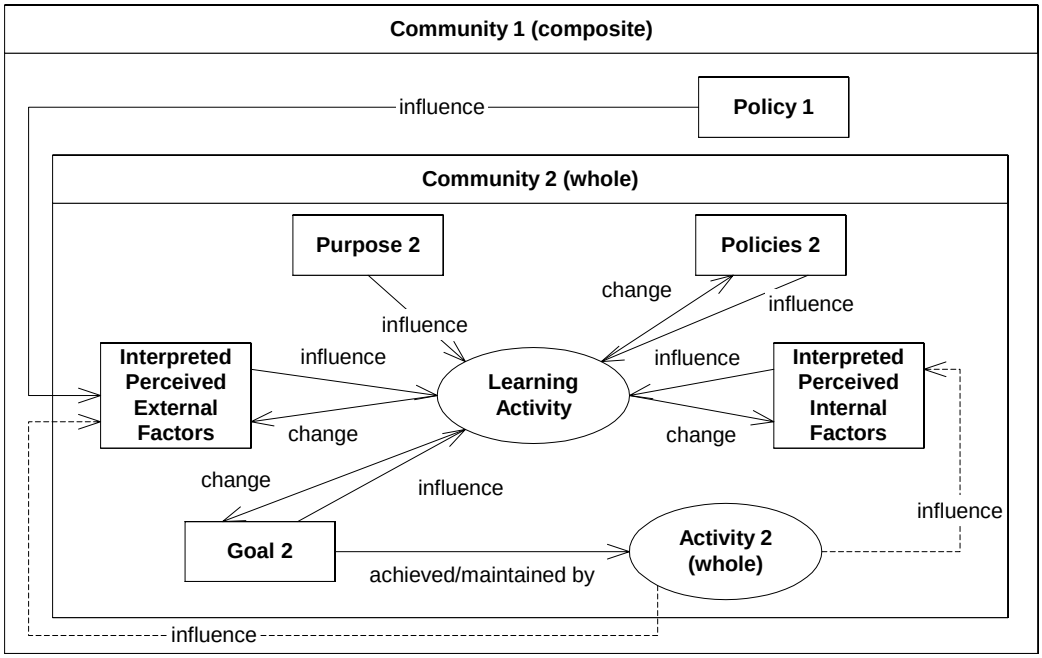

Figure 1. A Community Represented as a Whole

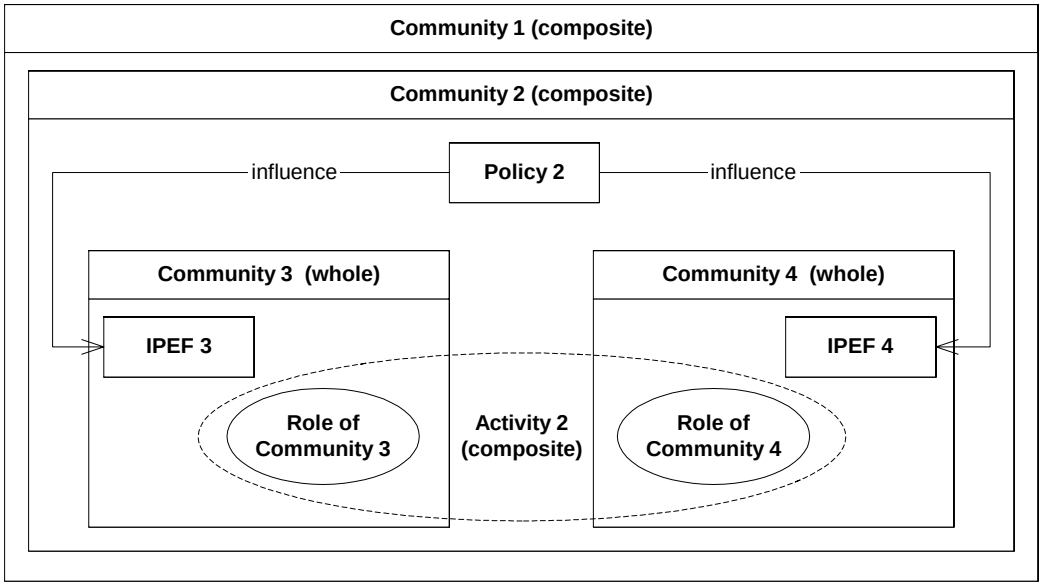

Figure 2. Community Represented as a Composite

Communities perform activities for a certain purpose or goal. It is possible to distinguish between two meanings of the concept of purpose [9], [27]: the function (or role) the community or activity serves within its supra-community and the purpose a community has set for itself. The first kind of purpose is what is generally referred to as role in software engineering [47]. The second kind of purpose is internally motivated and generally specifies the long-term maintenance of a state of affairs aimed at preserving the autonomy of the system. This specifies an activity that is dependent on the lifecycle of the community, i.e. it starts with the inception of the community and ends at the death of the community. This purpose can be

operationalized into goals that can be expressed in terms of the state of affairs of the community [27]. Goals that specify a state of affairs to be reached within a certain time frame (i.e. after an activity is performed) are called achievement goals. Goals that specify a state of affairs to be maintained during a certain timeframe (i.e. while an activity is performed) are called maintenance goals [2], [3].

Communities have relationships with other communities through the activities that they perform together. Communities influence each other through these relationships. In our framework we model these influences by using the concepts of "perceived factors" and "interpreted perceived factors". A perceived factor describes the influence of a community on a community of interest, as the community of interest perceives it. This perception is usually a somewhat distorted view of the reality of the community. The sensing mechanism of the community of interest may change some stimuli, fail to record some stimuli or record some non-existent stimuli. Following this perception, the community of interest interprets the perceived factor by making some judgment about its importance and how it affects its purpose and goals. The interpretation of the perceived factors is in itself a complex activity, which we will not attempt to explain. However, we can model the result (Figure 3), which is sometime called emergent properties, i.e. properties that the community views as a result of its relationships with other communities as defined by Weinberg [48, p. 60]. These emergent properties may be interpreted by the community as constraints, as capabilities and as both constraints and capabilities. Constraints represent obstacles to the achievement of a goal or purpose. Capabilities represent conditions that enable the achievement of a goal or purpose. As an example, consider a meeting, which can be seen as a tool for the goal of solving a problem (capability) or as the need to define and assemble the participants (constraint) or as both at the same time.

Constraints and capabilities influence the definition of policies. A policy can be seen as the response of the community to the constraints and capabilities it interprets as being subject to. In other words, a policy is a plan that enables the community to achieve its goal or purpose, taking into account the set of constraints and capabilities that the community believes it has. A policy helps with the refinement of the goal or purpose into sub-goals, which are than carried out by sub-activities.

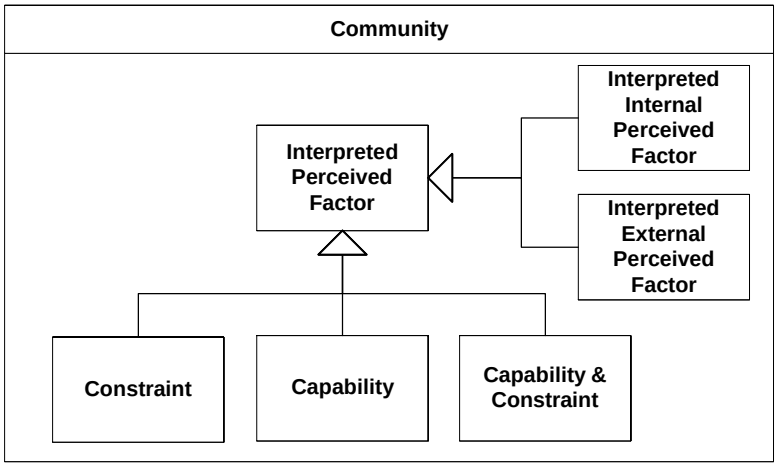

Figure 3. Interpretation of Perceived Factors

Note: the empty arrowhead represents sub-typing. Thus a capability is a sub-type of Interpreted Perceived Factor.

A perceived factor can be interpreted as an external or an internal factor. We call "Interpreted Perceived External Factor" (IPEF), a perceived factor that is interpreted as being due to the relationships of the community of interest with one or more communities that are external to its boundary. We call "Interpreted Perceived Internal Factor" (IPIF) a perceived factor that is interpreted as being due to the relationships of the community of interest with some of its sub-communities.

For reasons of brevity, we will refer to both Interpreted Perceived External Factor and Interpreted Perceived Internal Factor with the term “Interpreted Factors” when we don’t need to distinguish between external and internal factors.

The distinction between internal factors and external factors is dependent on what we consider to be the boundary of the community. When modeling physical systems (such as a computer) it is easier to define the boundary than when we model human organizations. In human organizations this boundary depends on the context that we model and its interpretation by the modeler. [9]. Thus, a consultant to a company may be seen as a member of a project community or activity (i.e. internal to the community) to a project manager and as a non-member of the company (i.e. external to the community) from the point of view of the payroll department. The distinction between factors that are internal to a community and factors that are external is based on the assumption that a community has more power to modify its internal factors than its external factors.

Our model of a community (Figure 1), inspired by Vickers [46], defines a succession of internal activities, which we call the learning activity. Through each occurrence of the learning activity, the community changes its goals, policies, interpretations and perceptions based on its purpose, its interpretation of the perceived factors (internal and external), and its present policies and goals. The community achieves its goals by performing the activities defined by its policies. These activities may change the interpretation of perceived external factors of the communities that have relationships with the community of interest thereby causing indirect changes to the interpreted factors of the community of interest (defining another occurrence of the learning activity). This defines a feed-forward (the goals, policies) and a feedback loop (interpreted factors). The interpretation of the perceived factors is also modified by the learning activity.

The cornerstone of our framework is the set of definitions given in Table 1. These definitions describe our understanding of the relationships among the concepts of community, activity, goal, interpreted factor (i.e. constraints and capabilities), policy etc. These relationships can help us understand the reasons behind goals as they are likely to be found in initial requirements documents such as scenarios, use cases, business rules, explicit policy statements etc. The idea is to understand that these goals proceed from community policies (that remain mostly implicit) that are dependent on the perceptions and the interpretations of the community. We can then represent the activity that is performed by the community of interest with other communities. Understanding the higher-level activity is crucial in reconciling the goals of the stakeholders in the different communities.

Organizational level	Level of perception of the observer. The level of perception defines the communities and activities that are considered as atomic by the observer
Emergent Property	A characteristic that a community perceives and interprets from its relationship with other communities
Boundary	An interpretation made by an observer of what differentiates a sub-community from an external community
Community	A model of something that is perceived as stable A community can be represented as atomic (i.e. a whole) at a specific organizational level or as a composite (composition of sub-communities) at a lower organizational level. A community performs a set of activities. Each activity is performed for a purpose or goal. A community has policies that define the roles of its sub-communities. The community has interpretations of itself and its relationships with other communities, which constrain or enable the achievement of its goals or purpose. These interpretations together with its purposes and goals shape its policies
Community of interest	The community currently under investigation
Supra-Community (of a community of interest)	A community within the boundary of which the community of interest (i.e. sub-community of the supra-community) exists
Sub-Community (of a community of interest)	A community that exists within the boundary the community of interest (i.e. its supra-community)

	A community performs a sub-activity within an activity of the community of interest. A sub-community usually subscribes to the purposes and goals of the community of interest
Member (of a community)	Synonym of Sub-Community
Perceived factor (of a community)	A state of affairs perceived by the community.
Interpreted factor (of a community)	An interpretation of a perceived factor by the community. An interpreted factor may be a constraint, a capability or both simultaneously. A perceived factor can be interpreted as an internal state of affairs (Interpreted Perceived Internal Factor) or as an external state of affairs (Interpreted Perceived External Factor)
Constraint (in a community with respect to a goal or purpose)	An interpreted factor that represents an obstacle to the achievement of the goal or purpose
Capability (of a community with respect to a goal or purpose)	An interpreted factor that enables the achievement of the goal or purpose
Learning Activity (of a community)	An internal activity through which the community changes its goals, policies, interpretations and perceptions based on its purpose, its interpreted factors and its present policies and goals
Activity (of a community)	A model of something which is perceived as happening An activity can be considered as atomic at a specific organizational level or as a composition of sub-activities at a lower organizational level
Purpose (of a community)	An internally motivated preference for the long-term stability of a state of affairs aimed at preserving the autonomy of the community
Achievement Goal (of a community)	A state of affairs of the community after the successful completion of an activity
Maintenance Goal (of a community)	A state of affairs the community maintains by performing an activity.
Role (of a sub-community)	The sub-activity that a sub-community performs within an activity of its supra-community as perceived by the sub-community
Policy (of a community with respect to a goal or purpose and a set of perceived factors)	A set of rules that defines the activities needed to achieve the goal or purpose taking into account a set of interpreted factors A policy is elaborated and changed by and within the community

Table 1. Definition of concepts

4 Discussion

Our framework is inspired in part by the writings of Sir Geoffrey Vickers [8], [46]. Vickers rejected the notion of individuals and organizations as goal-seeking entities. Vickers views them instead as maintaining a set of relations with regard to a certain value or norm. Vickers's model of organizations is far different from the "traditional" model used in the information systems field, that of organizations as goal seeking entities. Checkland argues that this traditional model is based on H.A. Simon's model of problem solving [39] which doesn't address the issue of goal definition, whereas in Vickers's model the main activity of the organization is the debate about possible courses of action [8, pp. 45-46].

Vickers defines norm as, [46, p. 14] "*governing relation by which the actual course of affairs may be judged*." Thus, for Vickers, individuals and organizations are self-regulating entities rather than goal-seeking entities. Goals are a by-product of their regulation rather than their main preoccupation. Accepting this view implies (in the goal-oriented vocabulary) giving more importance to maintenance goals than to achievement goals. This has been noted by Anton and Potts [3] who say that, "*An achievement goal is satisfied when the target condition is attained. A maintenance goal is satisfied as long as its target condition remains true. Maintenance goals are usually high-level goals with which associated achievement*

goals should comply." For example a manufacturing department is usually preoccupied with maintaining an acceptable quality level. This means specifying achievement goals when the quality drops below this acceptable threshold (or norm). Achieving a quality goal that cannot be sustained is not very useful in most cases. A maintenance goal defines an on-going activity that maintains the set of relations with regard to the norm. In the performance of this activity the community is influenced by factors that it interprets as constraints and capabilities prompting it to define achievement or maintenance goals, as well as plans of action for carrying these goals in the form of explicit or implicit policies

We call "learning activity" that which is performed within a community during which the community modifies its goals, its policies and the way it perceives and interprets reality. In other words, the learning activity of the community modifies what the community tries to achieve, as well as what it is capable of sensing and understanding. Our notion of learning activity is close to what Vickers [46, p. 16] calls "*appreciation*." Our concepts of perceived factors and interpreted factors resemble what Vickers calls the "*appreciative system.*" Thus Vickers [ibid] defines the appreciative system as the, "*set of readinesses to distinguish some aspects of its [an agent's] situation rather than others and to classify and value these in this way rather than that, and constantly to revise these readinesses.*" Vickers calls these a system because they function as a whole. A change to one of these entities (e.g. goals, policies etc.) automatically affects the others. Thus, what is perceived is dependent on what is interpreted as important. What is important is dependent on what is perceived. The appreciative system evolves historically (i.e. it is the product of past experience) and is dependent on the nature and organization of the community. Vickers [46, p. 20] gives the example that, "*A local authority is unlikely even to notice issues to which it is not organized to attend*; *and since its departments are designed for action, its appreciative capacity may be grossly limited or distorted by its organization*." In other words, a community's nature and the way it has evolved shape the issues that it will perceive and the way it will interpret these issues.

In our model, the interpretation factors (appreciative system) are linked with the policies and goals (what Vickers calls "*the readiness to do*") through the learning activity. Thus a policy is based on interpreted factors but cannot be changed without changing the interpreted factors that its definition was based on in the first place. For example, a manufacturing department will have an appreciative system that mainly perceives and interprets factors related to manufacturing, obscuring its role in the overall activity of the organization. Its policies and goals will reflect its interpreted factors. These policies and goals, which remain largely implicit, will be embedded in its practices and show up in scenarios, interview transcripts, etc. When an information system will be designed for this community and thus they will be the goals and policies for which the information system will be designed. The resulting information system, because of the lack of an overall view, may very well constrain the activities (or goal achievement) of the other communities in ways that are unacceptable to them. This is because the community may gradually consider its goals increasingly important and tend to view the other communities with which it performs activities as constraints and/or capabilities, that is, as obstacles or enablers on the achievement of its own goals [46, p. 33]

To go beyond this goal/constraint opposition and design information systems that satisfy all the communities involved means that we need to view the goals of the different communities with respect to the activity that they perform together. Activity Theory (AT) holds that individual human actions cannot be understood without reference to a minimal, non-arbitrary context [20]. Activity Theory defines an activity as a community of subjects (which we call members) who transform some object towards an outcome. The community members use tools for this transformation and are subject to rules managing (mediating) their relationships with the community and some division of labor (defining the roles within the activity). The rules, however, are not invariants and do not necessarily impose goals on the members. Activities are modeled by software engineers as "joint actions" [16] or as "collaborations" [31].

Human communities are autonomous entities. As defined by Maturana and Varela [26, p. 48], "*a system is autonomous if it can specify its own laws.*" Thus, a community can change its goals and policies with regard to its interpreted factors making it impossible to use the rules mechanically for inferring sub-community goals. For example, in the face of a rule that specifies that members of a community must attend meetings to which they are invited, some members may opt to find ways of avoiding these meetings. Identifying that some members do not participate in these meetings, the community may develop a meeting scheduling system that identifies free slots in members agendas, i.e. working even harder to get them to

attend the meetings they are trying to avoid. This may have the unwanted effect of these members trying even harder to avoid the meetings by, for example, keeping their agendas off-line.

Viewing the activity as a context for their goals may help the different communities to adapt their appreciative system to understand the goals and constraints of the other communities. These goals and constraints may stem from other activities the other communities are performing. Vickers also notes that the appreciative system can only sustain a certain rate of change before serious problems arise. Because goals and policies are linked to the appreciative system, they should also be changed with great care.

Our framework is based on Vickers's model with the addition of the concept of activity and more focus on the definition of goals. This may seem strange since Vickers thought that the focus on goals was misguided. However, we feel that defining goals for an information system to support is good practice. With our framework we can find the rationale for goals without focusing on them and without assuming a goal seeking organizational model.

5 The CompuSys Example

This example involves a case study of a failed internal product development within a computer manufacturer. The case was published by Markus and Keil in an article titled "*If We Build It, They Will Come: Designing Information Systems That People Want to Use*" in the Sloan Management review in summer of 1994 [28]. Markus and Keil identified most of the problems of this case using a Business Process Reengineering (BPR) framework. We offer an alternative interpretation based on our framework, which brings forward the influence of soft issues on stakeholders' goals more explicitly than in Markus and Keil's approach. This has the benefit of enabling stakeholders and engineers to reason on implicit policies and interpretations thus enabling them to find the rationale behind their design decisions. Furthermore, our framework should be easier to use in conjunction with other goal-oriented methods. Applying our framework to Markus and Keil's case we explicitly model activities, policies and interpreted factors. Thus, we show that the different communities involved in the project had different goals due to their differing interpretations of external policies; the policies of their supra-community (the organization) and the policies of their customers. We also show that these communities lost touch with the activity that they were performing together. Understanding this activity could have reconciled their goals.

Markus and Keil's article studies the real case of a computer manufacturing company they call CompuSys. CompuSys manufactures and sells custom configured computers for its customers. CompuSys is faced with quality problems stemming from the large number of different configurations available to customers. CompuSys sales reps often specify wrong configurations, which results in non-functioning computers being delivered to customers. Markus and Keil describe the different steps taken by CompuSys over the years in an attempt to solve this problem, none of which have succeeded. Markus and Keil's paper focuses on one of these efforts that consisted in the development of a configuration management software, called CONFIG, built by CompuSys to insure that configurations specified by the sales reps (who were to be the user of the system) were complete and accurate. Even though CONFIG was a "*textbook redeployment project*" involving costly user interface studies and training programs, it was never used extensively, or hardly at all, and the configuration problem was not solved. Markus and Keil show that the root cause of the non-use of CONFIG wasn't its user interface, lack of management attention, or lack of training but a bad, or non-existent analysis of the goals of the stakeholders and principle users of the system, i.e. the sales reps. However, the sales reps were heavily involved in the definition of the requirements for CONFIG. The problem was that the interpretation factors of CONFIG's developers prevented them from taking the most important needs of the sales reps into consideration.

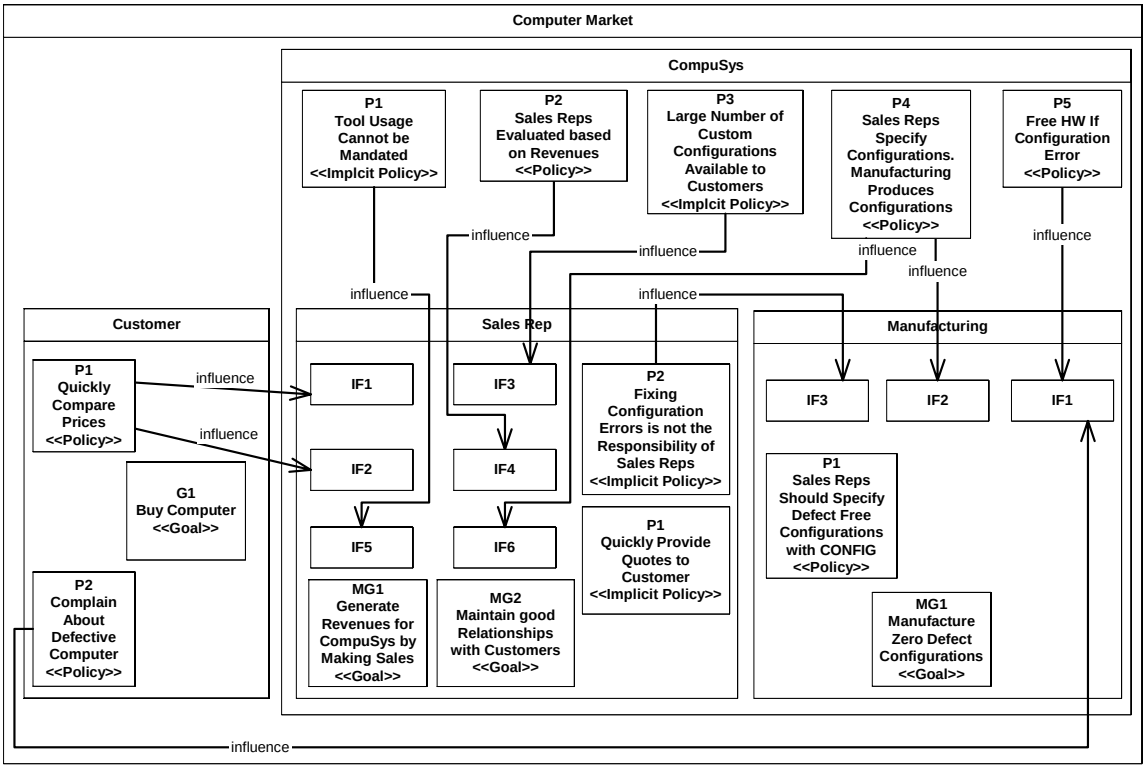

Figure 4. Explicit and Implicit Policy Influences within CompuSys

Figure 4 models the implicit and explicit policies, the interpreted factors and goals of CompuSys, its customers, the sales reps and the manufacturing department. In this model the relationships between the policies and the interpreted factors are represented in bold. We can thus see from these more or less causal chains that the manufacturing department, and the sales reps had seemingly conflicting goals.

Five CompuSys policies can be identified, that influenced the interpretation factors of the sales reps and the manufacturing department:

- Implicit policy: Tool Usage Cannot be Mandated (CompuSys.P1)
- Explicit policy: Sales Reps Evaluated based on Revenues (CompuSys.P2)
- Implicit policy: Large number of custom configurations available to customers (CompuSys.P3)
- Explicit Policy: Sales Reps Specify Configurations. Manufacturing Produces Configurations (CompuSys.P4)
- Explicit Policy: Free HW If Configuration Error (CompuSys.P5)

CompuSys.P4 specified that the manufacturing department's activity was to produce configurations which were specified by the sales reps. The free hardware policy (CompuSys.P5) and the customers' complaints about configuration errors (Customer.P2) made the manufacturing department aware of the high cost represented by the configuration problem (Manufacturing.IF1 and Manufacturing.IF3). These two interpreted factors represented the core of the problem, i.e. each occurrence of the defective configuration represents a high cost and there are many such occurrences. The manufacturing department was further influenced by its interpretation of the sales reps' inability to specify correct configurations (Manufacturing.IF3) and by what it thought the role of the sales reps was (Manufacturing.IF2), that is, specifying configurations. In summary, we can say that the manufacturing department viewed the situation in the following way:

- Maintenance goal: Manufacture zero defect configurations (Manufacture.MG1)
- Interpreted factor: Configuration defects have a high cost (Manufacturing.IF1)
- Interpreted factor: Sales reps role is to deliver non defective configuration specifications to manufacturing (Manufacturing .IF2)

- Interpreted factor: Sales reps often specify defective configurations (Manufacturing.IF3)
- Policy: Sales reps should specify defect free configurations with CONFIG (Manufacturing.P1)
- Implicit Policy: CONFIG should only manage configurations, not quotes (Manufacturing.P2)

CompuSys.P4 specified that the activity of the sales reps was to specify configurations for customers. The sales reps were influenced by the customers' policy of quickly comparing price quotes of different configurations (Customer.P1). This is shown in the model as SalesRep.IF1and SalesRep.IF2. In addition, the sales reps were not influenced by the CompuSys free hardware policy but by three other CompuSys policies, two of which were implicit policies:

- CompuSys.P1 influenced SalesRep.IF5
- CompuSys.P2 influenced SalesRep.IF4
- CompuSys.P3 influenced SalesRep.IF3

The sales reps viewed the situation in the following way:

- Maintenance Goal: Maintain good relationships with customers (SalesRep.MG1)
- Maintenance Goal: Generate revenues for CompuSys by making sales (SalesRep.MG2)
- Interpreted factor: Customers need several quotes to balance cost and performance (SalesRep.IF1)
- Interpreted factor: Price quotes need to be quickly communicated to customers (SalesRep.IF2)
- Interpreted factor: It's time consuming to specify a defect free configuration (SalesRep.IF3)
- Interpreted factor: Goal of sales reps is to generate revenues (SalesRep.IF4)
- Interpreted factor: Sales reps can't be forced to use CONFIG (SalesRep.IF5)
- Interpreted factor: Sales reps role is to specify configurations and provide their price quotes to customers (SalesRep.IF6)
- Implicit Policy: Quickly provide quotes to customers ((SalesRep.P1)
- Implicit Policy: Fixing configuration errors is not the responsibility of sales reps ((SalesRep.P2)

Two main imbalances can be identified in the way the CompuSys and the customer communities interact in this model. Customer sales are handled by sales reps but complaints about defective configurations are directed to the manufacturing department. Sales reps specify configurations for customers but are not evaluated on the basis of the quality of these configurations (Customer.P2 and CompuSys.P5 don't influence sales reps).

.

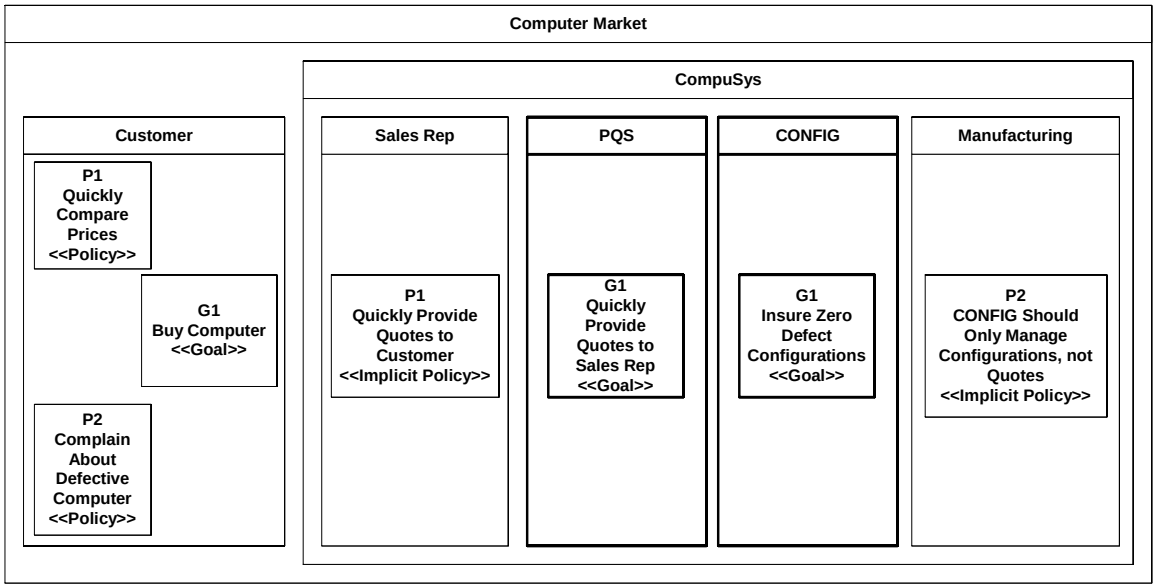

Figure 5. CompuSys Failed Solution (CONFIG)

Figure 5 shows the solution implemented by CompuSys, i.e. CONFIG. The CONFIG project was driven by the manufacturing department. Because of Manufacturing.P1 (Sales reps should specify defect free configurations with CONFIG), the intended users of CONFIG were the sales reps. The main high-level goal assigned to CONFIG was to create correct and complete configurations in accordance with Manufacturing.P2 (CONFIG should only manage configurations, not quotes). Thus, CONFIG was not designed to support the sales reps policy of quickly providing quotes to customers (SalesRep.P1). Markus and Keil note that the sales reps were using another program (called PQS) that was designed for their goals of quickly providing price quotes for configurations but which didn't verify the correctness of the configuration. So in order to provide a quote to a customer using CONFIG, the sales reps needed to specify a complete configuration, generate a configuration file and import it into PQS. This process conflicted with the sales reps policy of quickly providing quotes to customers (SalesRep.P1). CONFIG also conflicted with SalesRep.P2, which led the sales reps to believe that fixing configuration errors was the role of the manufacturing department rather than theirs. Thus, the sales reps viewed CONFIG as an obstacle rather than a capability. The sales reps implicitly invoked CompuSys.P1 through their interpretation of SalesRep.IF5 that stated that they could not be forced to use CONFIG if it didn't help them in their job. So they simply didn't use CONFIG.

Manufacturing.P1 was so implicit that Markus and Keil note that in spite of repeated requests by sales reps to integrate PQS and Config, the developers made this integration into a non-goal. They initially argued that integrating CONFIG and PQS was technically difficult, but as stated by Markus and Keil, the main obstacle was their view that CONFIG was a "stand-alone system" showing that their appreciative system just couldn't interpret this request as important.

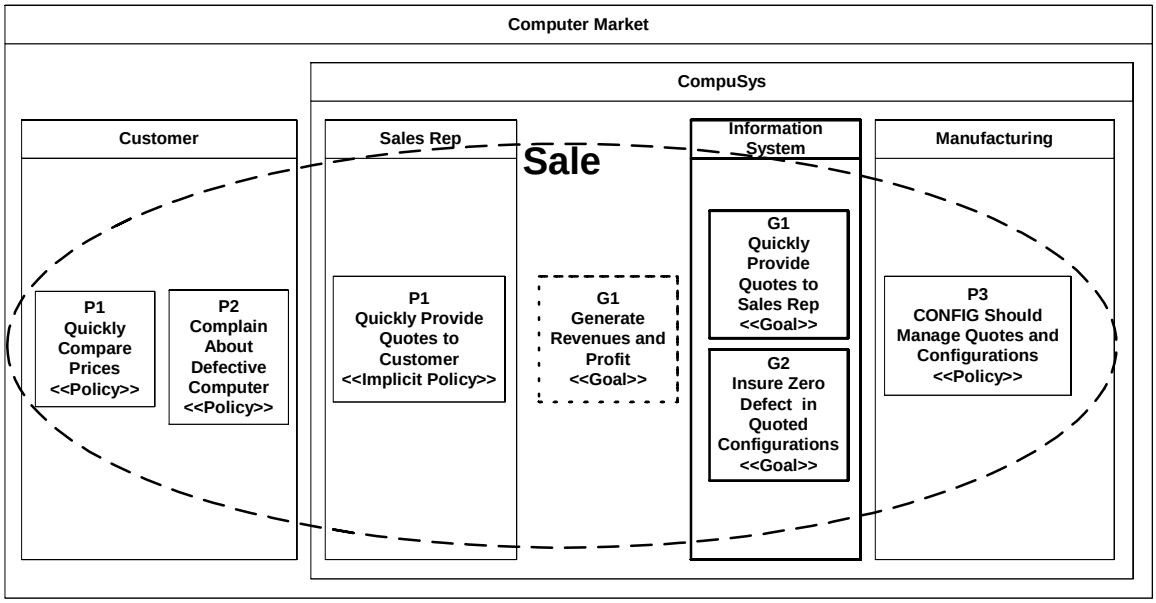

Figure 6. Activity Analysis for CompuSys

CompuSys should have realized that the sales reps, manufacturing department and customers were all involved in the sale activity. What was needed was a solution to make the sale activity more efficient. For that, they could have changed any of the following policies:

1. CompuSys.P1, i.e. mandate the use of CONFIG.
2. CompuSys.2, i.e. make sales reps responsible for configuration errors
3. CompuSys.P3, i.e. reduce the number of possible configurations to a smaller number of standard configurations
4. CompuSys.P4, i.e. change the role distribution between the sales reps and manufacturing
5. CompuSys.P5, i.e. adopt a less costly quality policy
6. Manufacturing.P2, i.e. redefine the high-level goals that CONFIG needed to support, taking into account the sales reps policies

Except for option 6, all the options above have major drawbacks. Options 1 and 2 would have slowed the sales reps and frustrated them. Options 3, 4 and 5 would have caused what Markus and Keil call: "*political and operational disruption out of proportion to the financial impact of the configuration error problem*." That is, they would have necessitated changing the appreciative system beyond its acceptable limit.

Option 6 shown in Figure 6, implied a change to the policies of the CONFIG developers alone. To do this, the developers should have viewed the activities of their department (manufacturing) as being performed within the larger sale activity. This sale activity is of utmost importance to CompuSys because it is the way by which the company generates the revenue and profit needed to keep its identity and autonomy (i.e. it is very close to its purpose). This view could have enabled the sales reps and the CONFIG developers to modify their partial interpretations of the sale activity. In the absence of an overall view, CONFIG developers used their implicit policies and interpretations to drive the project. They believed that involving the users in the project and defining the best user interface, followed by extensive training would make their envisioned solution a success. Thus, CONFIG is a great example of a very good product that was optimized for the wrong set of goals.

The high-level goals for the information system shown in figure 6 should themselves be refined into requirements for the information system. More implicit policies can be found in the sales reps way of carrying out their activities and these policies could be used to specify sub-goals for the information system or they could be changed because the introduction of the information system makes them obsolete or because they are deemed obsolete for other reasons such as organizational changes.

To conclude the CompuSys example, our framework offers a modeling approach that stakeholders and engineers can reason with in a given situation, where they believe that an information system is required. In our view stakeholders and requirements engineers need to build a model describing the concepts in Table 2.

The communities that interact and the activities that they perform together
The communities' policies (implicit and explicit) and their goals if available
The interpretation within each community of the other communities' policies in terms of constraints and capabilities, leading to conflicting goals
Higher-level activities which can be used to Reconcile conflicting goals
Goals that can be attributed to the information system in order to help the communities to perform their activities

Table 2. Necessary Modeling Elements

6 Conclusions

In this paper we described a framework designed to help with the definition of the set of goals for which enterprise information systems should be specified. Our concrete contributions are:

- The definitions of concepts in Table 1
- The modeling elements in Table 2

Note: Our notation represents several organizational levels within the same model. This is beneficial as shown in the CompuSys example. This kind of notation is not usually found in modeling languages such as UML.

Unlike current goal-oriented methods, our framework is not built on the model of organizations as goal seeking entities. This allows us to take an alternative view that we feel is more applicable to enterprise information systems where most of the issues surrounding requirements engineering are "soft" issues stemming from activities, implicit policies and their interpretations by different communities. Making these issues explicit enables stakeholders and engineers to reflect on the problems the organization is facing and on their belief that an information system is needed. Our framework does not make an assumption about the existence of an information system or the necessity of building one. Thus freeing requirements engineers from the necessity to think in terms of building or improving such a system.

Our framework is not intended to replace established, goal-oriented RE methods such as KAOS [14], NFR [29], GBRAM [2], ESPRIT CREWS [38] and the like. Rather, it aims at providing a community and activity based context for these methods, enriching them with the more complex issues encountered in an enterprise setting. Thus, our framework complements traditional goal discovery methods by providing aids for understanding the origin of goals.

Implicit policies and interpreted factors are difficult for people to state in explicit terms, which makes it difficult to apply our framework. As noted by Potts and Newstetter, [33] requirements engineers themselves have interpretations of what is needed and how to solve the problem. However, in order to avoid creating inadequate information systems, much effort needs to be invested for engineers' and stakeholders' implicit policies and interpreted factors to be made explicit. Contextual Inquiry [6] was designed for exactly the purpose of identifying, classifying and interpreting these soft issues through detailed on the job interviews. Beyer and Holtzblatt argue that these issues cannot be made explicit through classical interviews or scenarios.

Zave and Jackson [50] state that, "*goals by themselves do not make a good starting point for requirements engineering*." They argue that when engineers are given a goal for a system to be built they may decide arbitrarily to change the goal if they think that it was defined too narrowly, too generally or that it is the wrong goal to be pursuing and that this freedom is inappropriate. We believe that Zave and Jackson are correct because goals alone do not provide a sufficiently good picture of the situation to be addressed and the reasons for the given goals to be what they are. The goals given by stakeholders in scenarios and interviews represent their interpretation of the way they perform their activities. These interpretations usually describe a personal and partial view of an overall activity shown by Markus and Keil [28]. Our framework helps engineers and other stakeholders to understand how these personal views fit with an overall activity. Focusing on this higher-level activity and on the personal view of stakeholders at the same time shifts the point of view from the, often narrow, view of stakeholders while not losing site of their preoccupations. This helps with the definition of a set of goals that is more suitable for the organization as a

whole while insuring that that these goals will not be out of touch with the needs of the stakeholders as in Zave and Jackson's example.

7 References

[1] A.I. Anton, "Goal Based Requirements Analysis," in *Proc. Second Int. Conference on Requirements Engineering*., ICRE '96, pp. 136–144, 1996.
[2] A. I. Anton, "*Goal Identification and Refinement in the Specification of Software-Based Information Systems*," Ph.D. Dissertation, Georgia Institute of Technology, Atlanta GA, 1997.
[3] A.I. Anton, C. Potts, "The use of goals to surface requirements for evolving systems," in *Proc. of the 1998 Int. Conference on Software Engineering*, pp. 157-166, 1998.
[4] A.I. Anton, J.B. Earp, C. Potts, T.A. Aspaugh, "The role of Policy and Stakeholder Privacy Values in Requirements Engineering," in Proc. *Fifth IEEE Int. Symposium on Requirements Engineering, pp. 138-145, 2001.*
[5] L. Von Bertalanffy, *General System Theory*. New York: George Braziller, 1968
[6] H. Beyer and K. Holtzblatt, Contextual Design: Defining Customer-Centered Systems. San Farancisco, CA: Morgan Kaufmann, 1998.
[7] P. Checkland, and J. Scholes, Soft System Methodology in action. Chichester UK: Wiley, 1990.
[8] P. Checkland, S. Holwell, Information, Systems and Information Systems, making sense of the field. Chichester, UK: Wiley, 1998.
[9] P. Checkland, Systems Thinking, Systems Practice. Chichester, UK: Wiley, 1999.
[10] A. Cockburn, Writing Effective Use Cases. Reading, MA: Addison-Wesley, 2000.
[11] L. Constantine, "Essential modeling: Use cases for user interfaces," ACM Interactions, vol. II.2, pp. 34–46, Apr. 1995.
[12] L. Constantine and L. Lockwood, Software for Use. New York: ACM Press 1999.
[13] A. Cooper, "Goal-Directed software design," Dr. Dobbs Journal, Sept. 1996.
[14] A. Dardenne, A. van Lamsweerde and S. Fickas, "Goal Directed Requirements Acquisition," *Science of Computer Programming*, vol. 20, pp. 3–50, Apr. 1993.
[15] R. Darimont and A. van Lamsweerde, "Formal Refinement Patterns for Goal-Driven Requirements Elaboration," in *Proc. 4th ACM Symposium on the Foundations of Software Engineering (FSE4)*, San Francisco, pp. 179-190, Oct. 1996.
[16] D. F. D'Souza and A.C. Wills, Objects, Components, and Frameworks with UML – The Catalysis Approach. Reading, MA: Addison-Wesley, 1999.
[17] I. Jacobson, Object-oriented Software Engineering: A Use Case Driven Approach, Reading, MA: Addison- Wesley 1992.
[18] I. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process. Reading, MA: Addison-Wesley, 1999.
[19] H. Kaindl, "A Design Process Based on a Model Combining Scenarios with Goals and Functions," IEEE *Trans. Syst., Man, Cybern. A*, vol. 30, no. 5, September 2000.
[20] K. Kuutti, "Activity Theory and its applications in information systems research and design," in *Information Systems Research Arena of the 90 's*, H.-E. Nissen, H.K. Klein and R. Hirschheim, Eds. Amsterdam: North-Holland, 1991, pp. 529-550.
[21] J-L Le Moigne, La modélisation des systèmes complexes. Paris: Dunod, 1990.
[22] N.G. Leveson, "Intent specifications: An approach to building human-centered specifications," IEEE *Trans. Software Eng*., vol. 26, pp. 15–35, Jan. 2000.
[23] P.F. Linington, Z. Milosevic, K. Raymond, "Policies in Communities: Extending the ODP Enterprise Viewpoint," in Proc. *2nd IEEE Enterprise Distributed Object Computing Workshop*, pp. 14-24, Nov. 1998.
[24] P.F. Linington, "Options for Expressing ODP Enterprise Communities and Their Policies by Using UML," in Proc. *Third Int. Enterprise Distributed Object Computing Conference, EDOC '99*, pp. 72 –82, Sept. 1999
[25] C. Marshall, Enterprise Modeling with UML: Designing Successful Software Through Business Analysis. Reading MA: Addison-Wesley, 1999.
[26] H.R. Maturana and F.J. Varela, The Tree of Knowledge. Boston, MA: Shambhala, 1998.
[27] J.G. Miller, Living Systems. Niwot, CO: The University Press of Colorado, 1995.
[28] L.M. Markus, M., Keil, "If We Build It, They Will Come: Designing Information Systems That People Want to Use," in *Sloan Management review*, summer 1994.
[29] J. Mylopoulos, L. Chung and E. Yu, "From Object-Oriented to Goal-Oriented," *Communications of the ACM*, vol. 42. No. 1, Jan. 1999.
[30] B.A. Nixon, "Dealing with Performance Requirements During the Development of Information Systems," in Proc. *IEEE Int. Symposium on Requirements Engineering*, pp. 42-49, 1992.
[31] OMG: Unified Modeling Language Specification. Version 1.3, 1999. [Online]. Available: http://www.omg.org
[32] K. Pohl and P. Haumer, "Modelling Contextual Information about Scenarios," *Third Int. Workshop on Requirements Engineering: Foundation for Software Quality RESFQ*, Barcelona, Spain, June 16-17, 1997.

[33] C. Potts and W.C. Newstetter, "Naturalistic Inquiry and Requirements Engineering: Reconcling Their Theoretical Foundations," in Proc. Third *IEEE Int. Symposium on Requirements Engineering, pp. 118-127, 1997.*
[34] J. Rasmussen "The Role of Hierarchical Knowledge Representation in Decision Making and System Management," *IEEE Trans. Syst., Man, and Cybern.* vol. 15, no. 2, Mar./Apr. 1985.
[35] ISO/IEC ITU-T: Open Distributed Processing – Basic Reference Model – Part 2: Foundations. Standard 10746-2, Recommendation X.902, 1995.
[36] ISO/IEC JTC1/SC33/WG7, "Information Technology –Open Distributed Processing - Reference Model – Enterprise Viewpoint," ISO ISO/IEC 15414 | ITU-T Recommendation X.9ff, August. 2001.
[37] S. Robertson and J. Robertson, Mastering the Requirements Process. Reading MA: Addison-Wesley, 1999.
[38] C. Rolland, C. Souveyet, C. Ben Achour, "Guiding goal modeling using scenarios," *IEEE Trans. Software Eng.*, vol. 24, pp. 1055–1071, Dec. 1998.
[39] H.A. Simon, Sciences of the artificial. Cambridge, MA: MIT Press, 1996.
[40] M. Sloman, "Policy Driven Management for Distributed Systems," *Journal of Network and Systems Management*, Plenum Press. Vol. 2 No 4, 1994.
[41] M.W.A. Steen and J. Derrick, , "Formalising ODP Enterprise Policies," in *Proc. Third Int. Enterprise Distributed Object Computing Conference, EDOC '99*, pp. 84-93, 1999.
[42] A.G. Sutcliffe and N.A.M. Maiden, "Bridging the Requirements Gap: Policies, Goals and Domains," *Proc. 7th Int. Workshop on Software Specification and Design*, Redondo Beach, CA, December 1993.
[43] A. van Lamsweerde, R. Darimont and P. Massonet, "Goal-Directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt," *2nd Int. Symposium on Requirements Engineering (RE'95)*, York, UK, pp. 194-203, March 1995.
[44] A. van Lamsweerde, "Requirements Engineering in the Year 00: A Research Perspective," in *Proc. 22nd Int. Conf. Software Engineering*, Limerick, ACM Press, June 2000.
[45] A. van Lamsweerde, E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering," *IEEE Trans. Software Eng.* Special Issue on Exception Handling, 2000.
[46] Vickers, Sir Geoffrey, Policymaking, Communication, and Social Learning, New Brunswick NJ: Transaction Books, 1987.
[47] A. Wegmann, G. Genilloud, "The Role of "Roles" in Use Case Diagrams," *Lecture Notes in Computer Science*, Berlin Heidelberg New York: Springer-Verlag, pp. 210 – 224, 2000.
[48] G. M. Weinberg, An Introduction to General Systems Thinking. New York: Wiley & Sons, 1975.
[49] E. Yu, J. Mylopoulos, "Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering," *Proc. 27th Hawaii Int. Conf. System Sciences*, Maui, Hawaii, vol. 4, pp. 234–243, Jan. 1994.
[50] P. Zave, M. Jackson, "Four Dark Corners of Requirements Engineering", *ACM Transactions on Software Engineering and Methodology*, pp. 1-30, 1997.