Requirements Information Management: The NATURE Approach[1]

Matthias Jarke, Klaus Pohl, Ralf Dömges, Stephan Jacobs, Hans W. Nissen

Informatik V, RWTH Aachen, Ahornstr. 55, 52072 Aachen, Germany
jarke@informatik.rwth-aachen.de

ESPRIT project NATURE aims at the development and evaluation of Novel Approaches to Theories Underlying Requirements Engineering. Such approaches are necessary because the environment in which RE has to operate has changed dramatically since the current methods were invented. The task of RE has moved from supporting the early phases of individual projects, to accompanying the whole lifecycle of complex, long-lived human-machine systems in a rapidly changing organizational environment. The ***NATURE framework*** *addresses these new demands by defining a novel framework based on the idea that requirements engineering is a continuous process of establishing visions of different stakeholders in a complex context.*

Around this framework, NATURE has developed three specific theories. The requirements ***domain theory*** *gives advice what context knowledge is relevant and how to organize it. The requirements* ***process theory*** *offers a unified process meta model in which a small set of building blocks covers a larger spectrum of process guidance strategies with more flexibility than other software process or workflow models. The* ***knowledge representation theory*** *aims at defining what domain and process knowledge to capture, and how to manage this knowledge using an effective mix of informal, semi-formal and formal representations.*

While companion papers in this volume address the domain and process theories, we first present the overall NATURE framework and its implications for research directions in RE, and then focus on the knowledge representation theory. Specifically, we discuss the implications of the framework for repository-based process-centered RE environments, and some extensions towards large-scale distributed requirements

---

[1] This work was supported in part by the European Community under ESPRIT Basic Research Project 6353 (NATURE). A version of this paper appears in the Special Issue on Requirements Engineering of the journal *Techniques et Sciences Informatiques*, Spring 1995.

management based on a generalization of the viewpoints idea. The concepts have been validated in a comprehensive prototype environment called PRO-ART which is also presented.

1 The Changing Role of Requirements Engineering

Many engineering and business disciplines are experiencing the need to understand better the application aspects of their processes, and to maintain this information over time and across traditional technical and organizational boundaries. Within software engineering, requirements engineering (RE) is responsible for maintaining the relationships between a system and its environment. However, the meaning of RE is changing. The starting point of ESPRIT project NATURE was therefore an analysis of where RE is now, what we need, and what we have to offer.

The classical RE cycle consists of the three steps *elicitation - expression - validation*. Traditionally, the choice of requirements methods and tools has been determined by a single expected major source of complexity in the product to be designed: function (e.g., structured analysis), data (e.g., entity-relationship), or time-sequencing (e.g., statecharts) [BRA 90]. Object-oriented analysis methods (in-the-small) and enterprise integration architectures (in-the-large) look at several of the above perspectives simultaneously but their formal relationships and technical support remain poorly understood.

Several implicit assumptions have accompanied much of the early work in RE:

- There exists a well-definable *problem* that can be clearly scoped, described (and subsequently solved) once and for all; once completed, the system remains largely stable. The structured methods developed in the 1970´s are prototypical of this assumption.
- The requirements specification forms the basis of a *contract* which is fulfilled by someone else who is normally not responsible for the subsequent use and maintenance of the system.
- Each problem is different from others. Therefore, successful systems development is an art.
- *Users* are typically computer-illiterate but domain specialists who tend to know well what they want and just need to be questioned the right way. There exist only a few specialists who understand computers and have to be implicitly trusted.
- *Methods* for the early phases are generalizations of methods for later phases in systems development; they stem from *computer science*. Structured methodologies went from structured programming via structured design to structured analysis. The same historical pattern repeats itself for the object-oriented approach.

The traditional approach was quite successful as long as the software market was immature, organizations and their software systems were relatively stable over time, and information technology was localized in islands of automation. None of these conditions hold any longer. The central role of computer specialists in RE is abandoned in favor of broader participation, not only from users and their managers, but also from other potential "victims" of computerized systems. Therefore, requirements definitions must include broader context models, as well as traceability of all design decisions back to the central system vision or to the laws, policies, and habits of specific stakeholders. In the NATURE project, this problem is partially addressed by the *NATURE domain theory* which is described in [MAI 94c].

Moreover, the traditional one-shot approach to RE is replaced by a requirements engineering practice which gains added value by putting together standard components in an easily adaptable fashion. This also has implications for the kinds of process models that can be used for RE -- a topic addressed by the *NATURE process theory* described in the contribution by Rolland in this volume.

The focus of this paper is the *NATURE knowledge representation theory*. It provides an integrated perspective relating the domain and process theories formally and through technically supported RE environments. In NATURE, we do not understand knowledge representation in the limited formalism-centric sense often used in artificial intelligence. In addition to a variety of modeling formalisms, the inclusion of domain and process aspects emphasizes the question *what* knowledge about RE products and processes needs to be represented, and *how* it can be captured, managed and used effectively in a teamwork process setting.

The NATURE knowledge representation theory starts with a simple but comprehensive RE framework, gained from a literature analysis with consideration of several theories of language and work. This framework is reviewed and partly justified in section 2. It naturally leads to a categorization of important research directions (not all covered in NATURE) which are summarized in section 3. In section 4, it is shown how the framework has been elaborated into a repository-based and process-centered requirements information management environment called PRO-ART which provides strong support for traceability-based change management. While the currently implemented version of PRO-ART follows an extended client-server approach, section 5 describes ongoing work within the NATURE project to scale the environment up to truly decentralized development, extending the idea of viewpoint-oriented requirements engineering.

2 The NATURE Framework for Requirements Processes

Conceptual support for change management is becoming a central task of RE. A look at one step in such a change process helps us to understand what this proposition implies. Figure 1 shows that propagating a change needs four steps: (1) abstracting a model from current reality, (2) defining the desired change within this model; (3)

implementing the changed model in the reality while (4) taking into account the existing context. Noticing the necessary juxtaposition of the vision created from the need for change, and the complex context in which it has to take place, NATURE has defined requirements engineering as *a process of establishing visions in context* [JP 93].

Many RE methods emphasize one of these aspects at the expense of the other. Top-down decomposition methods such as Structured Analysis do not say how context influences decomposition, whereas bottom-up approaches (ethnographics is an extreme example) fail to say how analysis can be focused on the vision.

One source from which a more comprehensive framework for RE can be derived is the class of products for which requirements are to be determined. In the sequel, we shall focus on information systems (IS) that control, manage, or inform about other systems for a particular usage purpose. We first provide a structure for the context of these systems, from it, we derive three dimensions for describing the process of RE.

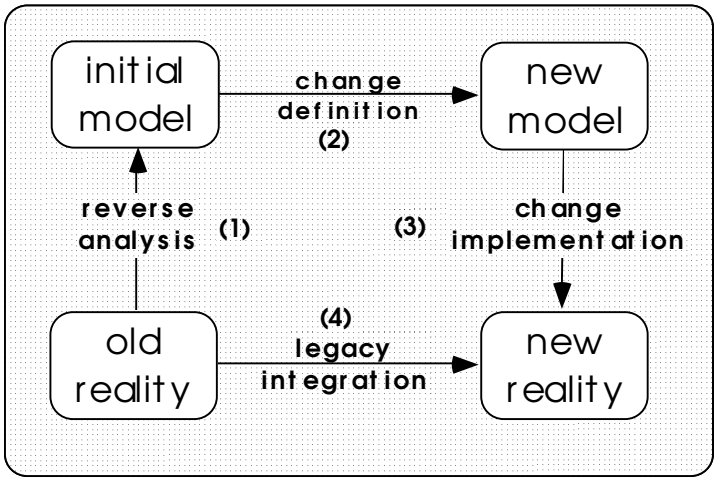

Figure 1: RE and the management of change

2.1 Structuring the Context

The context in which the system vision has to be established is complex and continuously changing. Establishing a vision in context remains an empty phrase if we do not understand what parts of the world are relevant and how they are related to the development process. We need a basic understanding of what RE is concerned with. To be acceptable to a broad class of developers, it must be easily understood and must not overly constrain freedoms in the RE process.

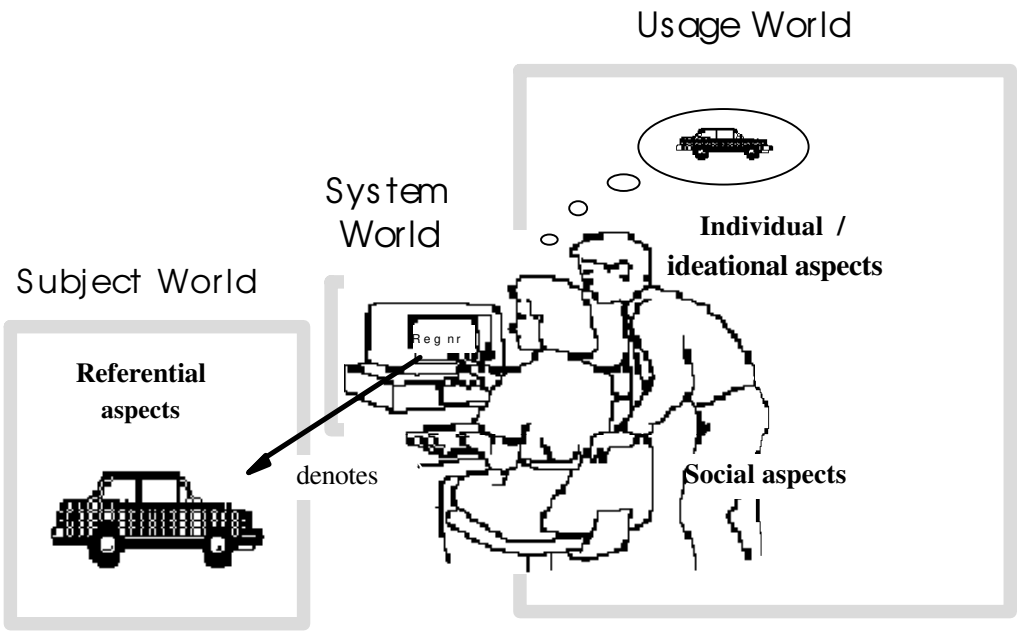

Figure 2: The operational context of an information system

The NATURE framework characterizes the context for IS by the three domains or "worlds" shown in figure 2 [NH 94].

First, the IS represents a certain part of the outside world which we call the *subject domain* — for instance, it may reference the domain of cars. Second, IS are designed with the purpose of being used by people, e.g. car buyers, with a particular goal in mind -- the *usage world*. We further distinguish individual, cognitive aspects of usage, from social or organizational ones. Third, the system itself needs some structured description or architecture; we call this the *system world*, e.g., a system that supports selecting a car.

The three worlds introduced above are neither static, nor do they cover the whole "real" world. Whether an object belongs to the usage, system or subject world (or even to two or three of them), is defined by the system vision, the goal of the change activity. Information about the subject domain is represented in the system after making a conceptualization. Depending on the system vision, this information may include information about the number plate of the car, its engine, the costs, the petrol needs etc. Using special system functions, again based on the development vision, the user is able to retrieve the represented information. Only if he is able to interpret the conceptualization and reflect it back to reality, the information system fulfils its cognitive purpose.

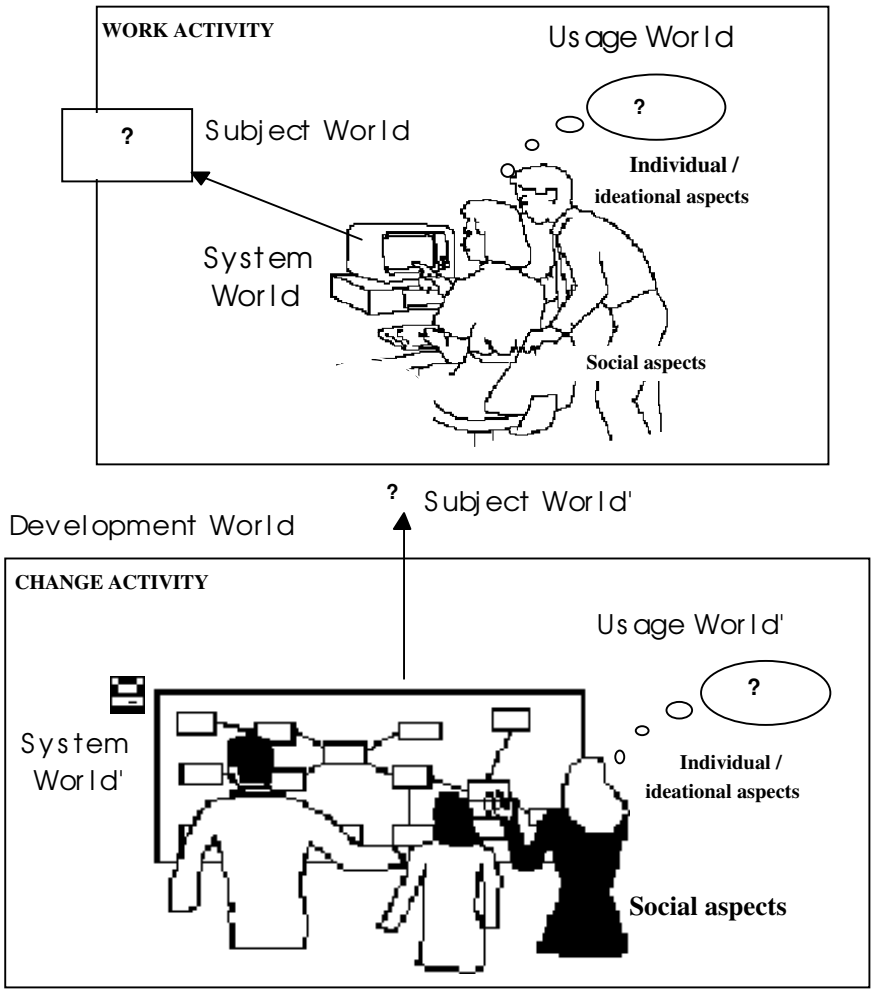

Figure 3 : The vision distinguishes subject, system, and usage world [NH 94]

For example, if the vision is to build a library system, books on the library shelves are relevant for the subject world. In an IS of a car repair shop, the same books could be part of the system world -- a part of the information system which represents cars for the usage purpose of repair. Hence, the vision guides the separation process that creates the three worlds from the infinite variety of real phenomena, and thus decisively drives the modeling process.

Figure 3 suggests to interpret the development world as an information system for the change activity. The *subject world* of our RE system is composed of information about the subject, usage, and system world of the target systems to be designed. Its *system world* comprises the technical support environment. The *usage world* of our RE system has to consider individual and social aspects of the RE team.

But how does the vision define the focus in practice?

Looking at RE from a social perspective, a vision holder establishes his or her vision in the social context through communication with other people, typically through

drawing them into a project team for certain periods of time. This team then becomes a distinguishable social reality of its own, drawing upon and interacting with the other three worlds. We call it the *development world*.

The need for change, and thus the seed for the development world, however, can originate in any of the worlds -- changing user needs or organizational environment, new technical possibilities in the system world or integration with another existing system, changes in the referenced reality, changes in an existing development team, changes in laws and regulations -- all of these could trigger an RE process.

Based on the context partitioning expressed by the three worlds, stakeholders in the RE process can be identified. They have different backgrounds and opinions due to the context they come from. It is important to deal with this background, in order to integrate the different aims and goals. Therefore, it is essential to gather information about the relevant parts of the different worlds, evaluate this information against the system vision, detect conflicts, and resolve them during the RE process.

During the RE process, knowledge about the context is structured in conceptual models or in instantiations of these models. This knowledge is acquired, conceptualized, and integrated within the development world, i.e. conceptual models about the subject, the usage, and the system world are developed.

Typical modelling formalisms exist for each world. Knowledge about the worlds may be already available in existing models; for example, business models provide a source for information about an enterprise and (depending on the vision) may be part of the usage world. Such models provide sources for reusing abstract information.

A description of the vision is then integrated into the conceptual model, i.e. the needed changes in the reality are identified and specified at the abstract level (cf. figure 1). During the design and implementation of the system based on this specification, the vision is integrated in the "real" context.

2.2 The Structure of the Development World

From the roles of the four worlds, we can predict certain goals they will likely impose on the system. For the development world, the model of figure 3 predicts three problem dimensions a requirements process would face in the development world (figure 4):

- The complexity of the individual/cognitive usage aspect leads to the *specification dimension* which describes the degree of completeness of the requirements specification, measured against some standard, guideline, or domain model. This dimension has traditionally come from practice, emphasized by large procurement agencies, but has recently seen substantial progress in terms of more

formal and detailed guidelines and reference models for various application domains.

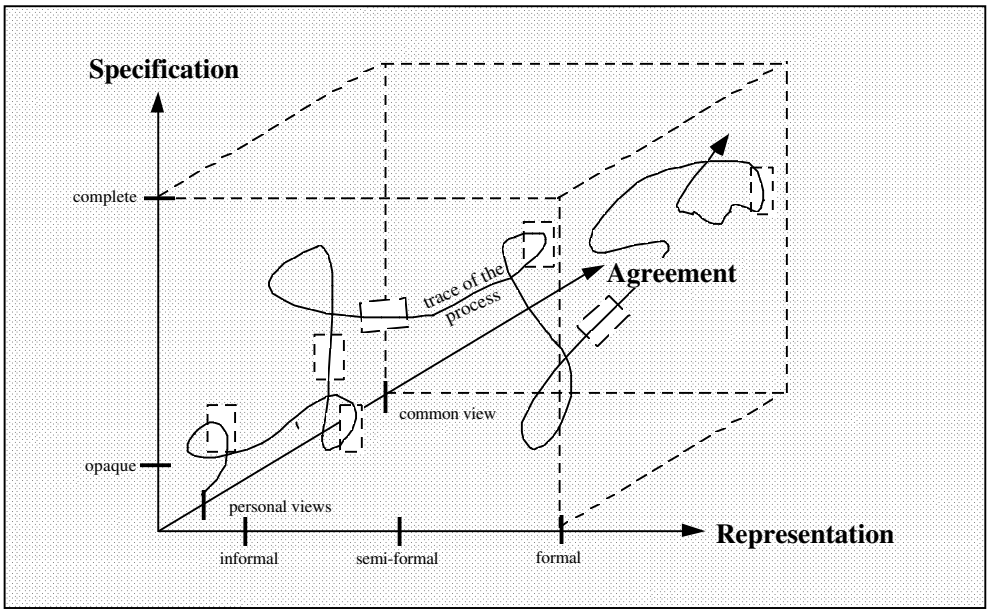

Figure 4 : The three dimensions of the RE process

- The social usage aspect leads to the *agreement dimension* which describes to what degree the members of the RE team agree on the requirements specification. This dimension has been investigated by social scientists.
- The system aspect leads to the *representation dimension* which describes how requirements are technically described, e.g. their degree of formal semantics. This dimension has mostly been studied by computer scientists.

RE processes can be described according to their effect on these three dimensions, for instance, transformation along the representation dimension, refinement along the specification dimension, or decision-making along the agreement dimension. The trace of the RE process is modeled as a curve within the three dimensions (cf. figure 4). Movement may sometimes appear erratic because the outside context of the other worlds changes. Moreover, as indicated by the boxes within the figure, only small chunks of this process are currently understood well enough so that decent guidance can be offered.

In [POH 93], these three dimensions have been independently derived from a thorough analysis of the published literature on RE. Pohl also showed that they have been studied by almost disjoint sub-communities. In NATURE, we are trying to bring these dimensions together to obtain coherent support for the RE process.

The three dimensions offer a way to trace or define the RE process. The most important demand on RE methods and tools is that they address *all* of these dimensions. In the NATURE project, we have first looked at individual requirements

for each dimension (see also [JAR 1993]), then focused on the question of integrated support.

Figure 4 shows that only certain pieces of the requirements engineering process are well enough understood to be supported by traditional software process models and workflow management techniques. Current software process models are not directly applicable as they assume broad coverage of the process by the models.

Process models in RE must first of all be *contextual*: at any moment in time, each developer is in a subjectively perceived *situation* upon which she looks with some specific *intention*. Situation and intention can change very quickly under control of circumstances that are only partially known. We therefore need a very flexible approach in which users can switch context at any time, and in which for any context whatever knowledge about the process exists, can be brought to bear.

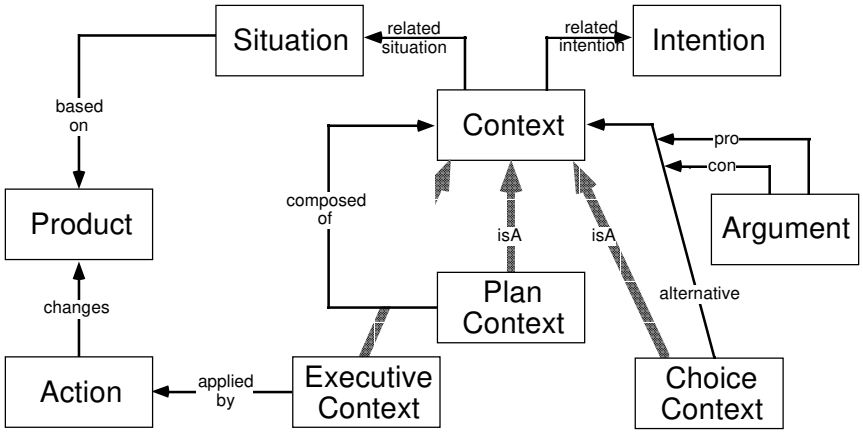

Figure 5: The NATURE process meta model

The NATURE process meta model (figure 5, cf. also [ROL 94]) achieves this flexibility by distinguishing three subclasses of contexts which can be nested arbitrarily (*composed-of* and *alternative* links in the model). *Executive contexts* describe immediate action that leads directly to product changes; they correspond to the traditional approach in CASE tools where developers just act without a formal computer-supported plan. A *plan* is a composite context that imposes a certain control structure over its part-contexts; this corresponds to the way software process modeling is looked at in most of the published models. Third, users often face a *choice* among multiple alternative contexts how to proceed further; the model associates pro and contra arguments with each alternative. Since contexts are recorded in the trace like products, dynamic process re-definition is also possible.

With a small number of constructs, the NATURE process model subsumes at least four earlier process modeling and guidance strategies. First, as in *decision-oriented* process models [JJR 90], intention and action are clearly separated from each other. Second, as in *situated* process modeling [RC 91], process knowledge is organized in individual chunks whose invocation sequence is not fixed in advance but based on the

situation partially determined by product status. Third, by simply attaching a specific kind of precedence structure (e.g., based on finite automata, statecharts, or Petri nets) to the *composed-of* link in the model, arbitrary *workflow* models can be supported. Fourth, the decomposition of a context in plan or choice context can be understood as an AND/OR graph -- the traditional means of decomposition-based *planning* in Artificial Intelligence [MCN 92]. To demonstrate this generality in practice, a number of RE methodologies, among them all major published object-oriented analysis methodologies, have been represented [GPS 94].

3 Research Implications of the NATURE Framework

The framework just presented has a number of implications for research and practice of RE which are discussed in detail in [JP94]. In this section, we briefly review these implications and relate them to work in the NATURE project.

Determining and understanding requirements is a hard task which is both too specific and too general to be attacked directly. It is too general because requirements come to life only through use case scenarios. It is too specific because each RE process is different and direct requirements reuse is rarely possible; metalevel architectures and domain models offer generic guidelines from which a requirements specification can be derived, or against which it can be validated. Thus, requirements teams must work outside-in, switching between scenario and meta mode to achieve the specification in the middle. NATURE has focused on the top-down part of this outside-in approach. The results are described for each of the three dimensions.

3.1 Coverage in the Specification Dimension

The specification dimension addresses the question of what a requirements specification should contain. With the shift away from a techno-centric view to the broader perspective indicated by the four-worlds model, the traditional facets of RE models have to be augmented with traceability to models of the context (system context, usage context, subject domain context, even development context), and to the vision and the goals of the various stakeholders.

In the sequel, we look at these two aspects in some more detail.

3.1.1 Domain modeling

The first organizations to consider the specification dimension were large procurement agencies who needed a standardized structure that made development proposals comparable. National and international standards organizations followed with standards and guidelines. In NATURE, we have conducted a comprehensive survey of twenty-one such standards and guidelines and condensed their main content into a conceptual model of about 40 classes with more than 100 attributes [GIB 94]. This model can then be used to compare the coverage of standards, to compare the contents of specifications organized according to different standards, to present a given

specification according to different standards, or to integrate requirements specifications of systems written using multiple different standards.

RSM - FUNCTIONAL REQUIREMENTS

Requirement \ Standard	IEEE-STD-830-1984	BS 6719, 1986	CSA 243.1-1979	FIPS PUB 101	ESA PSS-05-0 Issue 1	SMAP DID-P100	SMAP DID-P200SY	SMAP DID-P200SW	SMAP DID P210	JPL D-4003	JPL D-4005	FIRS PUB 38 FRT	FIPS PUB 38 SYS	DoD 2167A	DI-MCCR-80023	DI-CMAN-80008A	DI-MCCR-80025A	DI-MCCR-80026A	DoD-7935A FRT	DoD-7935A SYS	Starts Guide Vol.I
Abstract Requirement																					
ID	+						o	o									-				
Name	+		o								o						-				
Description	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Creator																					
Date of Creation																					
Relation			o				o	o		o				o			o		o	o	o
Priority	+	+			+		-	-		o				o				-	o	-	+
Stability	+																				
Acceptance Criteria		+		-	+	-				o			-				o				+
Alternatives		+																			
Abstract-Function-RQ																					
Inputs	+	+	o				-	-		o	-	o			-		o		+	+	+
Source	+	+	o								o						o	-		+	
Volume	+	+								o		o					o		+	+	
Format	+	+								o		o					o	-	+	+	
Range	+	+								o		o					o	-	+	+	
Errors			o																		
Outputs	+	+	o				-	-		o	-	o			-		o		+	+	+
Target	+										o						o	-		+	
Volume	+	+								o		o					o		+	+	
Format	+	+								o		o					o	-	+	+	
Range	+									o		o					o	-	+	+	
Accuracy		+	o							o		o	-				o	-	+	+	
Errors	+																				
External Interface	+	+	o	-	o	-	o	o	-	o	o	o	-	-	-	o	o	o	+	o	+
Time Behaviour																			+		+
Description of External System	+																				
User Interface	+	+			o	-			-		-					o		o	+		+
Type	+																				
Monitor Layout	+																				
Language		+																			+
Hardware Interface	+		o		o	-			-		-					o		o	+		+
Software Interface	+		o		o	-			-		-	o				o		o	+		
Functional Requirement	+	+	o		o		-	-		o		o									+
Processing	+	+	o		o		-	-		o					-						+
Validation of Inputs	+		o									o									
Operations	+	+					-	-													
Subordinate Functions	+																				
Validation of Outputs	+		o									o									
Error Handling	+	+																			o
Error Tolerance																					
Errors	+		o					o		o									+		+
Data Description			o			-	-	-		o		+					+	o	+		+
TBD	+				+					+	+										

"+" detail description; "o" described; "-" only mentioned; " " not mentioned

Table 1: Functional coverage of 21 requirements engineering standards [GIB 94]

As an example of this work, table 1 shows a table in which the coverage of the standards with respect to functional requirements is shown. It can be seen that only a few of the standards have a reasonable comprehensive coverage (the situation is even worse for nonfunctional requirements or business context requirements). Moreover,

these standards lack the detail to be truly useful as a guidance, and are purely geared towards what Lubars et al. [LPR 93] have called *customer-driven RE.*

As the software market matures, we see a trend towards low-cost standard packages for which a traditional RE process is too expensive. For individualization of such packages, we need easy *do-it-yourself RE methods*. They typically consist of a configuration of package functionalities, organized according to the business process to be supported. RE can be supported by reference models for various application domains, and each such reference model will offer various options for mapping its individual steps into meaningful configurations. The first environments along these lines are appearing on the market but lack formal foundations.

The NATURE project has therefore developed a formal theory of domain modeling [MS 93]. It comprises a library of about 150 subject world domain abstractions and 25 system world domain abstractions which have been empirically validated by laboratory studies and are supported by prototypical domain matching tools -- see also [MAI 94c].

3.1.2 Non-functional goals

While there has been some progress with metrics that *measure the achievement* of nonfunctional goals such as reliability, performance, accuracy, or security, we need to know more about how to use them in the requirements and design process.

One proposed solution is again derived from a computer science concept, namely, the idea of planning with AND/OR graphs [MCN 92]. Goals are broken down into subgoals until known satisficing solutions for achieving a subgoal are reached. Goal trees can be used to represent general knowledge about how to reach a certain goal, or to record specific requirements refinement processes. As mentioned in section 2, this kind of AND/OR decomposition is also supported by the combination of *composed-of* and *alternative* decompositions in the NATURE process model.

An alternative approach generalizes the idea of Essential Systems Analysis [MP 84] to deal with domain-specific knowledge about individual types of goals [DDD 94]. It starts with an unconstrained functional description of the intended system, derived from the vision. Then, the methodology takes each goal into account in a particular sequence, by gradually considering ways how it could be violated. For example, in a distributed system, the initial version would assume fully reliable components and communication lines. Then the methodology would investigate if one or the other is in fact likely not to be reliable and, should that be the case, to add requirements for extra measures that ensure reliability anyway.

3.2 Broadening the Agreement Dimension

Social aspects of RE have been a concern for a sub-community within research which has proceeded rather independently from the specification and representation activities. In our opinion, these aspects are part of a common whole and need to be investigated as such. For example, if you want to use ethnographic methods in RE, you

still need to keep the general goals of RE processes in mind -- achieving a sufficiently precise and agreed specification of the envisioned change with limited resources in a limited time [SRS 93]. Conversely, when devising a new formalism, reasoning technique, or process model, its social aspects in terms of understandability, learning effort, or political acceptance need to be considered.

The formal treatment of goals has to be linked to their social meaning. This also implies that domain modeling must cover usage world aspects such as organizational modeling. In the NATURE project, a model is being developed which identifies the information system as a medium in organizational speech acts [HOL 93].

The four-worlds model indicates that we should broaden the participation in RE teams. User participation has been a long-standing concern of RE but remains a difficult issue even just from the viewpoint of the question who are the "users" to be involved. More importantly, many systems nowadays have impact beyond their usage and development environment. Technically, this impact tends to be related to the "subject world", e.g., the patients in a hospital information system, or the neighbors of a chemical plant to be controlled by the system under design. Ways must be found to help these stakeholders (who are neither domain specialists nor technical specialists!) with analysing the system impact from their own perspective.

Moreover, the range of group support functions needs to be extended:

- *group knowledge building and maintenance:* Team members have different backgrounds and change frequently over time. The RE environment should assist in rapid transfer of acquired knowledge to newcomers, but also stimulate mutual learning of the team as a whole.
- *conflict management:* Conflicts of many different kinds -- ranging from technical concurrency control to contract violation handling, from formal negotiations to hardly structured argumentations -- need to be recognized and dealt with in RE. Technical means help mostly in the detection, rarely in the resolution of these conflicts. The ex-post analysis of such conflicts is useful as a rationale for design decisions and their possible revision over time.
- *traceability:* It is often important to trace requirements back to their origin, for example, to determine their criticality. It is equally important to assign *accountability* for certain requirements to identifiable team members [RE 93].
- *acceptance of guidance:* Regardless of whether guidance in the RE process is provided by oral advice, handbooks, or technical means, it is important that arguments help the user decide among possible solutions or work strategies; theory says that argumentation should be grounded in identifiable experiences.

Technically, these different functionalities should be available in a cooperative work setting which includes synchronous as well as asynchronous, local as well as remote, message-passing as well as repository-oriented modes of cooperation.

3.3 Diversity in the Representation Dimension

The representation dimension of RE has traditionally been characterized by the debate of informal vs. formal. Computer science wisdom says that the final requirements specification should be "formal", but in practice it is most often not. Indeed, there is evidence that informal information dominates all the way down to the implementation level. However, it is also the question whether 3.000 folders of "informal" text and graphics can be understood by humans.

The relevant goals of representation are *human intuition and communication,* and *computer-supported analysis.* This is not achieved in a single formalism but in a well-chosen combination of intuitive textual and graphical representations with formally understood and automatically captured interactions. In the NATURE project, especially adapted hypertext structures with an explicit underlying formal model have been developed to support this combination [HAU 94].

The main impediments on the way to this goal are complexity of requirements specifications, and incompatible specialist languages in the different worlds.

3.3.1 Complexity handling with abstractions

The explication of the basic abstraction principles of classification/ instantiation, aggregation/ decomposition, and generalization/ specialization in the mid-1970´s has had a profound impact on RE methodologies. While the early formalisms (structured analysis, entity-relationship, statecharts) just rely on decomposition, more recent proposals also consider the other two.

Originally proposed by [BGM 84], stepwise specialization has become an important requirements development strategy in object-oriented analysis. Working with reference models implies *bottom-up reuse and composition of solution objects* rather than *top-down refinement of requirements*. This strategy can also be observed in other engineering disciplines. Engineers tend to think from the available components, their behavior, and their interaction. The actual goals and global functions of the system are just a derived property, not the starting point. Because the properties of solutions are better known in advance than those of possible problems, this approach does not only facilitate reuse but also the preparation of animations and other presentation media useful for human intuition.

This brings us back to the central role of the classification dimension already alluded to in the beginning of section 3. On the one hand, the standardization of knowledge in the specification dimension (section 3.1) emphasizes the need for a meta level representation which proposes the schema for individual requirements projects. On the other, the need for human intuition and communication (section 3.2) requires examples and scenarios which actually instantiate the schema under development. In the NATURE project, we chose Telos [MBJ 90] as a formal requirements language that -- invisible to developers and other stakeholders -- manages the relationships between meta level domain schema, actual specification, and usage scenarios of this specification.

3.3.2 Multiple requirements perspectives

The discussion about the specification dimension showed that a requirements specification involves many different facets. Traditional tools concentrate on one of these facets and offer a set of closely related representations around it. Object-oriented analysis is a first step towards a broader spectrum of facets. However, interrelating these different views without massive training and tool support has proven difficult.

Moreover, the assumption that there is at any given moment a coherent "object" from which the views are derived, forces early design decisions concerning the grouping of objects. This may be acceptable in simple cases where RE will become synthetic rather than analytic, as discussed above. But it is arguably incorrect in the presence of multiple stakeholders with different opinions, and in complex analysis tasks where a lot of learning takes place. We therefore expect that truly distributed approaches to RE will prevail over object-oriented analysis.

The price for the autonomy and realism of distributed requirements engineering is the need for managing the relationships among viewpoints. Meta modeling of static interrelationships has now reached a degree of maturity that commercial CASE environments support a rough name-based mapping between most of the well-known representations. To integrate the informal legacy documentation, it is important to also structure informal objects with limited costs.

Even more important, but less understood, are the *dynamic dependencies* among RE perspectives. For example, the relationship between a formal representation and an informal text could be of type *derived-from, example-for,* or *opinion-about* -- in either direction! These dependencies result from the process followed in RE. They can only be recorded during process execution, hardly reconstructed afterwards.

Static and dynamic relationships can be partially predicted by, and tested against, the meta level product and process models. The rest still needs substantial manual recording; reducing this effort by concentrating on just the relevant relationships and automating the recording is a major challenge for RE tool developers and method designers. The question when and how to invoke automatable checks also depends on human factors such as the interruption of work, awareness of parallel change, and the like. Solution proposals come from the groupware field (group awareness in user interfaces) and the active database community (detached trigger handling).

4 Implications for Process-Centered Engineering Environments

The contextual approach followed by the NATURE process meta model allows us to consider the process model as a set of more or less independent process chunks associated with the white pieces of figure 4. However, since we know little about the RE process, even with such a flexible model, the traditional process-centered environments will not offer much support. More importantly, they contribute nothing to the acquisition of *new* process knowledge from collected experience. To improve on

this, the RE environment must also capture and support the activities where no process definition exists.

In this section, we first use a framework proposed by Dowson to explain more clearly the deficits of existing approaches, and to define the requirements for a solution. Then, we describe a prototype system called PRO-ART which addresses these requirements through a combination of repository technology, process-integrated tool concepts, and advanced process services [POH 94].

4.1 Extending the Principles of Process-Centered Environments

Dowson has summarized the three principal components of a process-centered environment, as shown in figure 6 [DOW 93]. The process modeling domain (PMD) is responsible for creating, storing and maintaining process models. The process enactment domain (PED) is responsible for the management of actual process instances. It retrieves instantiated process definitions from the PMD, and controls the actual process performance through a process engine that enacts the instantiated model. The process performance domain (PPD) actually performs the engineering activity, supported and controlled by the PED. The PPD also gives feedback to the PED, a prerequisite for having branches, backtracks, or loops in the process.

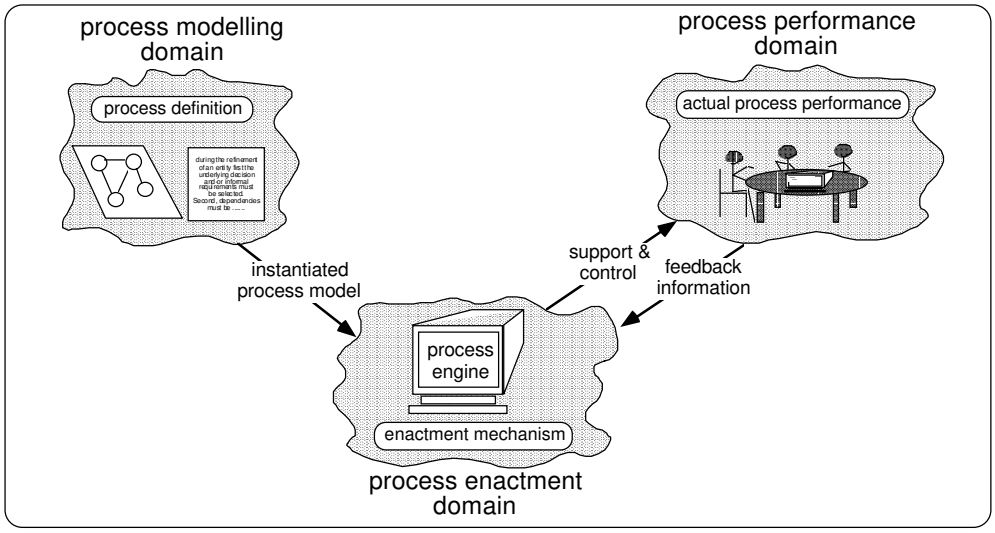

Figure 6: Information flow in current process-centered environments

Most commercial environments have very weak links between the PED and the PPD. Developers work autonomously on individual tools and are loosely linked to the PPD by some kind of reporting system. In contrast, proposals in the software process community as well as most commercial workflow management systems assume that a process model covers the process completely, and enforce it rigidly.

This assumption is not justified in RE. The approach breaks down as soon as we leave a defined area (either finish or violate a defined process), nor does it support recognition of the fact that the process has entered such an area.

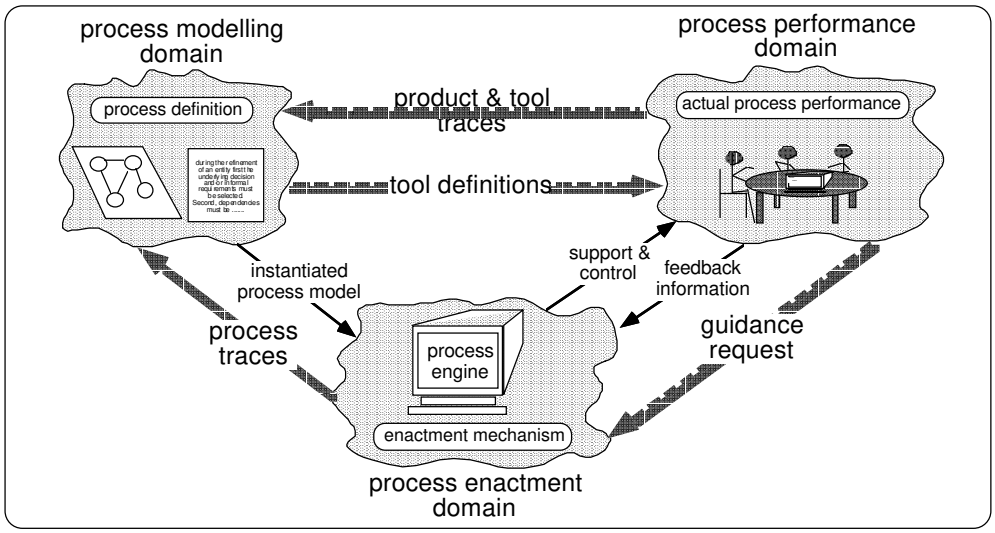

Figure 7: Information flow in process-adaptable and traceable RE environments

These considerations inspired the extension of the model shown in figure 7. It adds four additional communication links to the three in figure 6:

- *guidance request:* The developer should be in the driver´s seat, rather than being controlled by the PED. If agents in the PPD believe that they are in a situation where a process chunk is available for proceeding further, they can ask the PED to retrieve and enact it.
- *process traces:* Since we cannot assume anymore that the PED is continuously active, it becomes very important that it leaves a trace of its activity in the process modeling domain. This trace can be used to determine the overall status of a distributed engineering process, but it can also be used for process improvement, as discussed later.
- *product & tool traces:* This is the most crucial extension. Outside the defined process chunks, the architecture in figure 6 offers no trace of the actual activity. Since most of the RE process is in the grey area, this means that no meaningful trace of the overall process is available and therefore nothing can be learned from it, either for project management or for process improvement. We therefore need a way to generate a meaningful trace from the products created and from the usage of tools that created them.
- *tool definitions:* Tool usage patterns are defined as process chunks so that any tool-supported activity in the PPD can be traced in terms of the process model. By sending tool definitions from the PMD to the PPD, it becomes also possible to change tool behavior in line with changing process definitions, thus incorporating

process-driven behavior in the tools rather than just in a separate guidance tool (such a guidance tool is still needed for visualizing the larger process context -- cf. [ROL 94]).

Even when a process chunk is active, the separate recording of the product and tool traces can be useful because it yields valuable information about process violations which could be the starting point for corrective action in the PED; note that this comparison of planned and actual process in the PMD is different from the direct feedback from PPD to PED in figure 6. In the long run, this information can also lead to improvement of the process definitions. Especially in the case of RE, a much better empirical understanding of the RE process should result from this approach, in turn leading to much better guidance; we have started empirical studies with a major CASE vendor to validate this claim.

4.2 Process-Centered and Repository-Based Architectures

We have shown that the process modeling domain needs richer patterns of communication with the PED and PPD. Figure 8 presents an architecture that satisfies these requirements [PJ 92, JPR 94].

The kernel piece of this architecture is a process repository. It coordinates the process definition environment (PMD), to the left of figure 8, with the process coordination and execution environment (PED and PPD), to the right. The repository follows the level organization of the IRDS standard [ISO 90] in that each level constitutes a type system for the level below.

The lowest level contains the traces which, in correspondence with figure 8, also include the products developed. For the defined pieces of the RE process, the traces are documented according to guidance models executed in the PED. For the whole process, product and tool traces are captured according to traceability models.

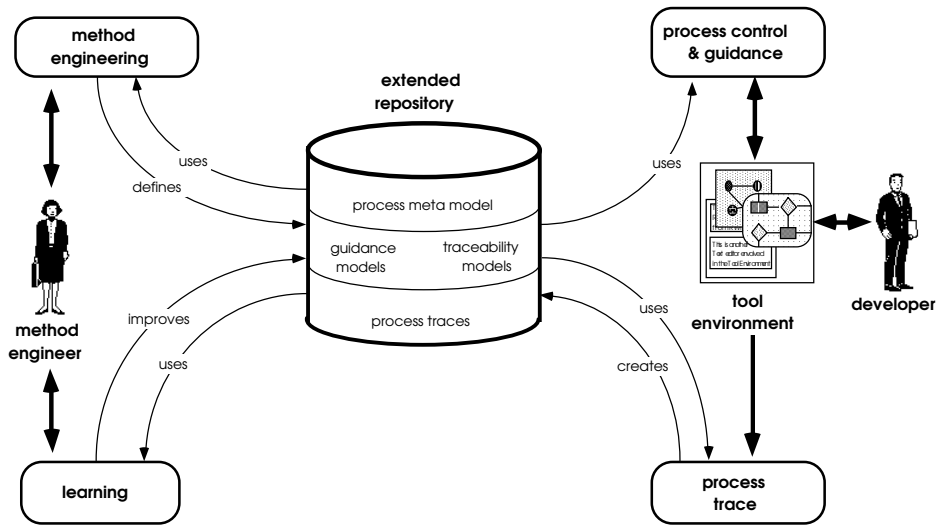

Figure 8: Repository-based and process-integrated environment concept [JPR 94]

The process meta model shown in figure 5 is a common meta model for traceability and guidance meta models in the PMD. Its concepts are interpreted somewhat differently in either case. For example, in the traceability submodels, the arguments associated with choice context *document* what arguments developers have actually stated in the process; in process guidance, they constitute generic decision support in certain kinds of situations.

The guidance meta model contains some additional constructs specifying process abstractions and control structures for plan contexts. Similarly, the traceability meta model contains additional information about the agents in the PPD and the dependencies among products created by the process.

As the links between the process repository and the two environments illustrate, the architecture is intended to work as follows. The PED (process control and guidance) retrieves guidance model chunks from the process repository which then influence the PPD (tool environment and developer) to follow the chunk. The actual performance is recorded automatically in the process trace according to the traceability models (and the guidance models used, if any). The prerequisite for this to work are tools implemented as an extensible and re-configurable set of services.

In the PMD, process evolution is supported as shown on the left of figure 9. Either, the process meta model can be instantiated with new guidance or process chunks from outside. Or a comparison between the traces and the guidance models used can lead to experience-based improvement [JPR 94].

4.3 The PRO-ART Prototype Environment

A prototype environment called PRO-ART was developed in the NATURE project to validate this architecture (figure 9). The process repository is managed by ConceptBase [JEG 94], a deductive and object-oriented database supporting the conceptual modeling language Telos [MBJ 90]. Telos offers object-oriented structuring mechanisms with an arbitrary number of meta levels to introduce extensible modeling languages, as well as logic-based deductive rules and integrity constraints across multiple such levels to define their semantics. This cannot be done with traditional database languages such as SQL.

Guidance and traceability models in PRO-ART were derived from the three dimensions described in figure 4, staying as close as possible to current industrial practice with the representations chosen. In total, these models comprise about 250 Telos classes; they are linked by 18 different types of dependencies. The user does not normally have to be aware of these models and dependency types since they are automatically created by the tool environment.

In the representation dimension, hypertext was chosen for informal representation, entity-relationship and structured analysis for semi-formal representation, and Telos itself for formal representation. The set of semi-formal representations is extended to the full set of OMT representations [RBP 91]. In the specification dimension, a

conceptual model of what should be in a requirements specification called RSM was extracted from twenty-one international standards and guidelines (cf. table 1 [GIB 94]). In the agreement dimension, an extension of the IBIS argumentation model [CB 88] defines a decision made in the process as a view on an evolving argumentation space.

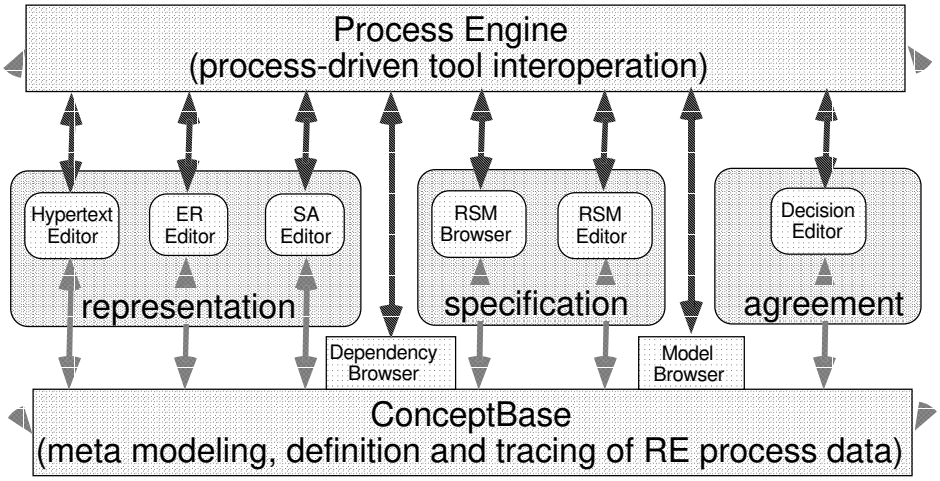

Figure 9: The PRO-ART system

Tools corresponding to the different models mentioned above are shown in the middle of figure 9. The behavior of each of these tools in terms of both traceability and guidance can be redefined from the corresponding process model in ConceptBase.

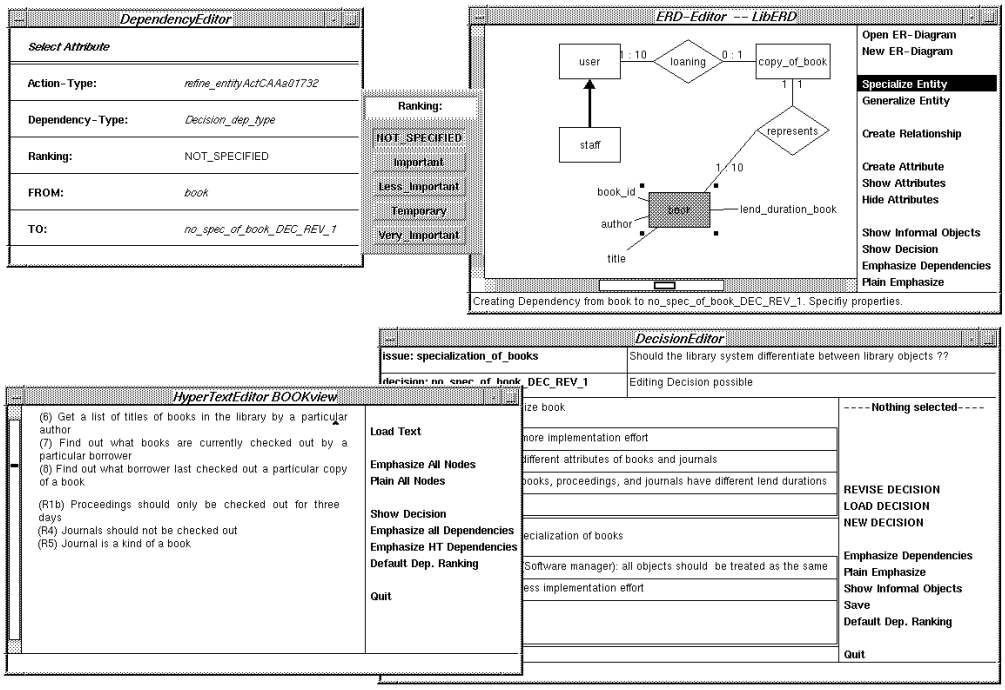

Figure 10: A PRO-ART screendump showing a multi-tool guidance chunk

The screendump in figure 10 illustrates a multi-tool guidance chunk which offers a special process for refining an entity in the *ER Editor*. This process first simply adds a menu item *specialize-entity* applicable to entity classes to the *ER Editor*. When this item is selected, however, it is first checked in the *Decision Editor* (started via the *Dependency Browser*) whether a decision relevant to this operation exists (plan context); if yes, the user can optionally look for informal justifications of this decision in the *Hypertext Editor* (again via the *Dependency Browser*, following a choice subcontext); further, the decision may have to be revised before returning to the *ER Editor* and actually doing the specialization.

The PRO-ART *Process Engine* drives such multi-tool processes without losing context [POH 94]. It is not a process interface to the user but an extended background communication server which retrieves a piece of process definition from the repository, and uses it to manage interoperability across the tools involved in this piece. It knows what tool instances and process chunks are currently active on the developer´s screen and takes advantage of this knowledge. For example, if a tool requested by another tool is already present on the screen, the Process Engine will find this tool and put it in the right mode according to the process status (i.e., the right objects and the right operations are focused on the screen). From other tools for communication integration (such as HP´s SoftBench), it is distinguished in that it preserves process context across several tool calls.

5 Towards Distributed Requirements Engineering

The opinion that software development is a principally distributed activity comprising multiple viewpoints gains increasing acceptance in the community [CKI 88, FKN 92, SRS 93]. Multiple stakeholders hold different perspectives on future or existing products; *no* correct central view fully unifies the representations used, the cognitive contents of the specifications, or the opinions about them. Instead, there is only partial and temporally brittle agreement about partially shared representations with only partially shared understanding of the problem. These relationships between viewpoints must be explicitly supported and maintained over time.

However, from the perspective of our three-dimensions model, research in viewpoints and viewpoint integration has been quite limited so far. Following long-standing work in database view integration [BLN 86, JOH 93], the focus has been on static relationships in the representation dimension [LEF 93, FGH 93]. Little work exists on dynamic aspects or the other two dimensions. In this section, we briefly review, for each dimension, some ongoing work in NATURE that aims at enriching viewpoint-oriented development.

5.1 Viewpoints in the Representation Dimension

From the representational perspective, a viewpoint can be defined as a loosely-coupled, locally managed object encapsulating knowledge about a system and its

domain, representation knowledge, the intended role of the viewpoint, a local trace of the RE process from its generation to the current state, process knowledge about how to further analyze and integrate the viewpoint, and knowledge about owners" [MST 94].

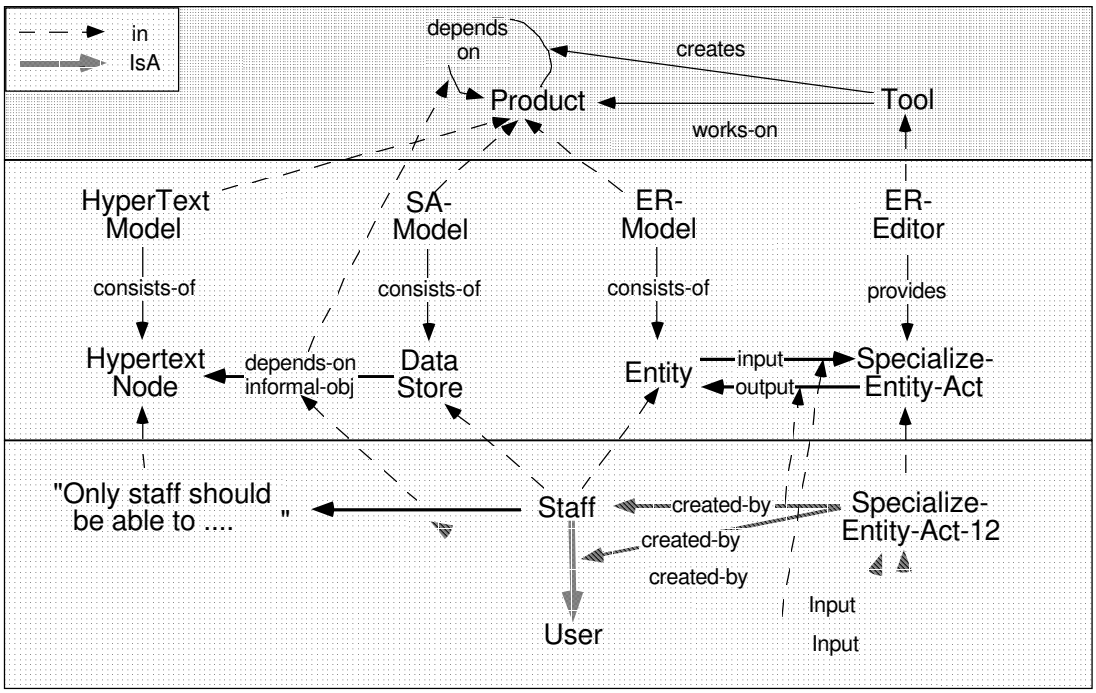

Figure 11: Static and dynamic relationships between representations

The starting point of our discussion is the repository-based architecture of figure 8. Figure 11 shows an example piece of the repository which relates hypertext, ER and SA representations of a simple piece of the library example. The purpose of this highly simplified figure is to illustrate how the current PRO-ART repository represents the meta model, its instantiation by different representations, the possible dynamic relationships between these representations created by tool action types, and the actual objects, tool actions and relationships created by them.

The figure shows two different kinds of integration among viewpoints:

- *Static integration* is achieved by multiple instantiation of product classes in the trace. For example, *Staff* is an instance of the representation classes *Data Store* in SA and *Entity* in ER. In the Telos language supported by ConceptBase, consistency of such multiple instantiations can be defined as follows. First, the process meta model concept of *Product* is specialized into a hierarchy of subclasses. The degree of similarity between product types (e.g., between the *Data Store* and the *Entity* concepts) is determined by the lowest common superclass of their meta classes. Second, each sub metaclass definition contains so-called meta constraints which constrain properties that instances of the instances of this subclass must have. These meta constraints are automatically

compiled into specialized constraints for each instance of the subclass [JEG 94]. Simple integrity checking at insertion time of product trace data will then ensure that multiple instantiation is only possible if the constraints from all classes are satisfied.

Above, we have described a passive means of integration through integrity checking. By using deductive rules instead of integrity constraints, immediate propagation of instantiations is also possible; for example, such a rule could automatically create a data dictionary entry in the SA model whenever a new entity is added to the trace.

- *Dynamic integration* is achieved in an analogous way through the tool models in our approach. Process or tool definitions are associated with the definition of dependency classes among product classes, and tool actions instantiate these dependencies. For example, the process model for entity specialization described in section 3.3 touches the hypertext editor and then creates a dependency link from the defined entity to the text used for justifying it. Thus, the principle of dynamic integration is the same as for deductive rules in the static approach. However, since the propagation is not automatic, dependency links have to be maintained explicitly at the trace level, whereas they can remain implicit for the static case.

  Consistency checking of the correct derivations is given by the 18 dependency classes defined in PRO-ART, and of course through the tool model definitions that have to use these dependency classes.

The enriched meta models are not only used for consistency checking, but also to define similarity in the search for reusable models [SC 94, MST 94].

Summarizing, PRO-ART simulates a static and dynamic viewpoint integration on a centralized repository. For a truly distributed approach, the architecture needs to be re-arranged. An active tool instance is associated with the corresponding pieces of the repository model into a viewpoint; a viewpoint then contains:

- a tool instance as the active component
- the relevant pieces of guidance and trace models
- the relevant pieces of trace
- knowledge about information exchange needs with other viewpoints
- knowledge about the consistency and agreement status of the viewpoint

Instance-level inter-viewpoint dependencies or object-sharing information are either handled in a central repository, or by separate "integration viewpoints". Local knowledge about exchange needs is in the form of triggers which can be partially generated from the metaclass hierarchy definitions mentioned under static integration.

A module concept has been added to the Telos language as a basis for defining logically distributed repositories and their interrelationships, check-in mechanisms for

the integration of multiple models in a team viewpoint have been dedveloped, and a specific meta model structure for similarity-based search has been selected [MST 94]. Additionally, a viewpoint status hierarchy has been defined which classifies a viewpoint according to the degree of internal and external consistency achieved, and with respect to visibility (private ... public) and acceptance (proposed ... generally accepted). Process models how to deal with recognized needs for communication have yet to be defined, dealing with issues such as where and when to propagate values automatically or just to send notifications, using eager or lazy propagating style, etc.

5.2 Viewpoints in the Specification Dimension

Meta modeling and multiple instantiation are also the technical basis for viewpoint definition and integration along the specification dimension. The guidance models should not just contain representational aspects of the process (e.g., whether a product is represented as an ER diagram), but also specificational aspects (e.g., whether the product must represent a physical object, an agent, etc.).

As discussed in section 3.1, NATURE has analyzed existing RE standards as a means of relating requirements specifications written in different standards. Moreover, the project has developed a library of domain abstractions which can be imported in a viewpoint to guide the local work (critiquing of proposed requirements models with respect to general knowledge about the domain). Such domain models can, however, also be used to guide the RE team in selecting suitable combinations of viewpoints, or in propagating the right information across viewpoints; for details, see [MAI 94].

5.3 Viewpoints in the Agreement Dimension

Viewpoint properties such as visibility and acceptance status refer to the agreement dimension of the RE process. Socially, a viewpoint can be defined as a social artefact produced as part of the RE process, in which its owners find shared meaning and through which these owners learn, communicate, and negotiate. An owner is any individual who can either change the viewpoint, directly or indirectly, or who is reponsible for the accuracy of its contents, or for the consequences of making public the viewpoint within its intended wider role in the project. The process from generating to completing the viewpoint is modeled as a single, interrelated series of activities undertaken by the owners. Each viewpoint is also private to its owners until it has explicitly been made public to other individuals in the process.

It makes sense to associate also goal structures with the social dimension of viewpoints. View integration therefore usually means *negotiation* which in turn consists of bargaining as well as mutual learning (compromise vs. consensus negotiation). In RE, problems in mutual understanding are especially frequent, as the background knowledge is necessarily quite different. In the NATURE project, we therefore investigate mechanisms of knowledge sharing across heterogeneous teams [PJ 94] as well as drawing on results from group decision support systems for negotiation. One such representation is the so-called House of Quality (HoQ [HC 88]) used in

industrial Quality Function Deployment to relate customer wishes, engineering options, and production management needs. In the simplest case, such a HoQ degenerates to a simple cross-reference matrix, as shown for our ER-SA integration example in the screendump of figure 12.

We have developed a toolkit for the construction of such viewpoint confrontation structures [GEB 94] which supports

- dynamic abstraction hierarchies within rows and columns, including the possibility for re-arrangement
- correlation within one viewpoint as well as across representations (representations within a viewpoint are represented as roofs)
- statement of individual as well as group preferences (represented as summary rows resp. columns)
- synchronization across different viewpoints; for example, if there are row summaries and column summaries for each owner, scrolling "around the corner" can be defined in order to make sure that always the information about the same owners is visible on the screen
- different real-time synchronization mechanisms across the Internet (same content/same representation, same content/different representation, with or without private pieces)

In addition to the decision-oriented trace model described in section 4, a guidance model for the agreement dimension across viewpoints has been defined for this tool, based on process models from the domain of concurrent engineering.

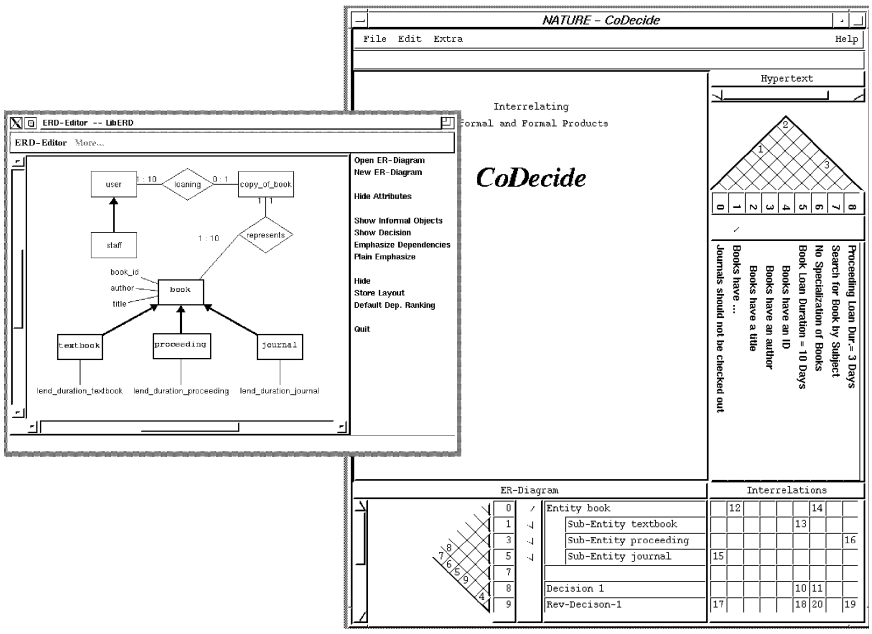

Figure 12: Confrontation of heterogeneous viewpoints in the "house of quality"

6 Summary

Requirements engineering is challenged to take a more important role in many domains, as its products -- metalevel domain models, conceptual specifications, and their instantiations in prototypical scenarios -- are becoming tools by which interdisciplinary teams can understand, communicate, and change reality.

A comprehensive understanding of the development process is critical to raise to the new application challenges. We do have some knowledge at the very fine-grained level of specific methods. There also exist basic approaches at the very coarse-grained level some of which were mentioned in this paper. However, in the middle, there is a big gap that can only be filled by careful analysis of existing and evolving practice. Not only to gain this bitterly needed experience, methods and tools for requirements traceability are central to our understanding what RE actually involves.

Despite progress with many individual problems, our overall understanding of the application engineering process is quite limited. RE has to overcome the current fragmentation of research interests along the different dimensions of interest, as well as the divergence between research and practice. This paper has therefore placed a major emphasis not only on making individual predictions about the future of RE, but also proposing a coherent framework within which these predictions can be positioned and combined. The specific demands of RE, explicated in the NATURE framework, lead to new concepts for process-centered engineering environments that enrich the communication pattern offered by current environments and also form a good starting point for distributed development strategies needed to support large-scale projects.

Acknowledgments. The concepts presented in this paper are embedded in an excellent collaborative effort of the whole NATURE team, whose other partners are led by Panos Constantopoulos (FORTH), Colette Rolland (Paris-Sorbonne), Alistair Sutcliffe (City University), and Benkt Wangler (SISU). Specifically, this paper has profited from the contributions of Peter Holm to the explanation of the four-worlds framework in section 2, of Jean-Roch Schmitt and his colleagues in Paris to the process model integration in section 4, and of Neil Maiden to definitional issues for the expanded viewpoints approach described in scction 5.

References

[BLN 86] Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys* 18(4), 1986.

[BMW 84] Borgida, A., Mylopoulos, J., Wong, H.K.T.: Generalization/specialization as a basis for software specification. In Brodie/Mylopoulos/Schmidt (eds.): *On Conceptual Modeling,* Sprinter-Verlag 1984, 87-114.

[BRA 90] Brackett, L.: CASE tools for requirements analysis and software design. Tutorial notes, *12th Intl. Conf. Software Eng.,* Nice, France, 1990.

[CB 88] Conklin, J., Begeman, M.L.: A hypertext tool for exploratory policy discussion. *ACM Trans. Information Systems 6,* 4 (1988), 303-331.

[CKI 88] Curtis, B., Krasner, H., Iscoe, N.: A field study of the software design process for large systems, *Communications of the ACM 31*(11), 1988.

[DDD 94] Derroitte, M., Dubois, E., Du Bois, P., Petit, M., Michel, J.-P.: Requirements definition of manufacturing systems. Working paper, Université de Namur, Belgium, 1994

[DOW 93] Dowson, M.: Software process themes and issues, *Proc. 2nd Intl. Conf. Software Process*, Berlin, Germany, 1993.

[FKN 92] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: a framework for integrating multiple perspectives in system development. Int. J. Software Engineering and Knowledge Engineering, 2(1), 1992.

[FGH 93] Finkelstein A., Gabbay, D., Hunter, A., Kramer J., Nuseibeh, B.: Inconsistency handling in multi-perspective specifications; *Proc. 4th European Software Eng. Conf.,* Garmisch, Germany, 1993, 84-99.

[GEB 94] Gebhardt, M.: Coherent design through viewpoint coupling (in German). Diploma thesis, RWTH Aachen, 1994.

[GIB 94] Gibbels, F.: A model for the specification of information systems (in German), Diploma thesis, RWTH Aachen 1994.

[GPS 94] Grosz, G., Plihon, V., Si-Said, S.: Describing various processes related to requirements engineering following the Paris I guidance meta-model, Deliverable, ESPRIT project 6353 NATURE, 1994.

[HAU 94] Haumer, P.: Structuring and integration of informal knowledge in requirements specifications by means of hypertext. Diploma thesis, RWTH Aachen, 1994.

[HC 88] Hauser, Clausing: The House of Quality. *Harvard Business Review*, July-August 1988.

[ISO 90] ISO-IEC 10027: Information Technology - information resource dictionary system (IRDS) - franework. ISO/IEC International Standard, 1990.

[JEG 94] Jarke, M., Eherer, S., Gallersdörfer, R., Jeusfeld, M., Staudt, M.: ConceptBase - a deductive object manager for meta data bases. *Journal of Intelligent Information Systems*, October 1994.

[JJR 90] Jarke, M., Jeusfeld, M., Rose, T.: Process management in the DAIDA environment. *Proc. 6th Intl. Software Process Workshop*, Hakodate, Japan, 1990.

[JP 93] Jarke, M., Pohl, K.: Establishing visions in context: towards a model of requirements processes. *Proc. 12th Intl. Conf. Information Systems,* Orlando, Fl, 1993.

[JP 94] Jarke, M., Pohl, K.: Requirements engineering in 2001: managing a changing reality. *IEE Software Engineering Journal*, November 1994.

[JAR 93] Jarke, M., Pohl, K., Jacobs, S., Bubenko, J., Assenova, P., Holm, P., Wangler, B., Rolland, C., Plihon, V., Schmitt, J.-R., Sutcliffe, A.G., Jones, S., Maiden, N.A.M., Till, D., Vassiliou, Y., Constantopoulos, P., Spanoudakis, G.: Requirements engineering - an integrated view of representation, process and domain. *Proc. 4th European Software Engineering Conf.*, Garmisch-Partenkirchen, Germany, 1993.

[JPR 94] Jarke, M., Pohl, K., Rolland, C., Schmitt, J.-R.: Experience-based method evaluation and Improvement: A process modeling approach. *Proc. IFIP 8.1 Working Conference: Methods and Associated Tools for the Information System Life Cycle (CRIS'94),* Maastricht, Netherlands, 1994.

[JJ 94] Jeusfeld, M., Johnen, U.: An executable meta model for re-engineering of database schemas. *Proc. Intl. Conf. Entity-Relationship Approach*, Manchester, UK, 1994.

[JOH 93] Johannesson, P.: A logical basis for schema integration. *Proc. 3rd Intl. Workshop on Research Issues in Data Engineering - Interoperability in Multidatabase Systems*, H. Schek (Ed.), 1993.

[LEF 93] Lefering, M: An incremental integration tool between requirements engineering and programming-in-the-large. *Proc. IEEE Symposium on Requirements Engineering*, San Diego, Ca, 1993.

[LPR 93] Lubars, M., Potts, C., Richter, C.: A review of the state of the practice in requirements modeling.*Proc. IEEE Symp. Requirements Engineering,* San Diego 1993.

[MAI 94a] Maiden, N.: Viewpoint resolution during requirements engineering -- a reuse-oriented approach. Submitted for publication, 1994.

[MS 92] Maiden N., Sutcliffe A.: Exploiting reusable specifications through analogy. *Comm. ACM 35*, 4 (1992), 55-64

[MAI 94b] Maiden, N.A.M., Sutcliffe, A.G., Till, D., Taylor, C., Nissen, H.W., Jarke, M., Pohl, K., Dömges, R., Assenova, P., Bubenko, J., Wangler, B., Johannesson, P., Spanoudakis, G., Constantopoulos, P., Halkia, K., Plihon, V., Rolland, C., Sisaid, S., Grosz, G.: Distributed requirements engineering within NATURE. Submitted for publication, 1994.

[MAI 94c] Maiden, N.A.M., Sutcliffe, A., Taylor C., Till, D.: A set of formal problem abstractions for reuse during the requirements engineering. Technique et Science Informatique, 1994 (this volume).

[MP 84] McMenamin, S.M., Palmer, J.F.: *Essential System Analysis.* Yourdon Press 1984

[MBJ 90] Mylopoulos, J., Borgida, A., Jarke, M., Koubarakis, M.: Telos: Representing knowledge about information systems. *ACM Trans. Inform. Systems*, 8(4), 1990.

[POH 93] Pohl, K.: The three dimensions of requirements engineerin. *Proc. 5th Intl. Conf. Advanced Information Systems Engineering,* Paris 1993.

[POH 94] Pohl, K.: *A Process-Centered Requirements Engineering Environment.* Dissertation, RWTH Aachen, 1994.

[PJ 94] Pohl, K., Jacobs, S.: Traceability between cross-functional teams. *Intl. Journal of Concurrent Engineering -- Research and Applications,* November 1994.

[PJ 92] Pohl, K., Jarke, M.: Quality information systems -- repository support for evolving process models. Aachener Informatik-Berichte 92-37, Aachen, Germany.

[RE 93] Ramesh, B., Edwards, M.: Issues in the development of a requirements traceability model. *Proc. IEEE Symp. Requirements Engineering,* San Diego 1993.

[RC 91] Rolland, C., Cauvet, C.: ALECSI: an expert system for requirements engineering, *Proc. 3rd Intl. Conf. Advanced Information Systems Engineering*, Trondheim 1991.

|ROL 94] Rolland, C.:Reformuler les demarches de conception . Technique et Science Informatique, 1994 (this volume).

[RBP 91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: *Object-oriented Modeling and Design*, Prentice Hall, 1991.

[SC 94] Spanoudakis, G., Constantopoulos, P.: Exploiting similarity between software artifacts. *Proc. 6th Intl. Conf. Software Engineering and Knowledge Engineering,* Jurmala, Latvia, 1994.

[SRS 93] Sommerville, I., Rodden, T., Sawyer, P., Bentley, R., Twidale, M.: Integrating ethnography into the requirements engineering process. *Proc. IEEE Symp. Requirements Engineering,* San Diego 1993.