PRO-ART: PROcess based Approach to Requirements Traceability*

Klaus Pohl, Ralf Dömges, Matthias Jarke
Informatik V, RWTH-Aachen
Ahornstr. 55, 52056 Aachen, Germany
Tel. +49/241/80–21513, Fax. +49/241 /80–21529
(Fax. +49/241/88–88321 from 01.02.94)
email.: {pohl, doemges, jarke}@informatik.rwth-aachen.de

Abstract: *Requirements Traceability is crucial for the success of system development activities. What does it take to capture, maintain and use requirements information? Based on a three dimensional framework and a tool interoperability approach we present a requirements engineering environment (PRO-ART) which enables the content oriented capture of requirements information. The demonstration shows how*

- *traceability can be used during the requirements engineering process;*
- *requirements information is captured during process execution by our tool environment; e.g., the interrelation of informal specifications and semi-formal notations like SA and ER diagrams;*
- *changes can be propagated in the current specification using the trace information.*

* This work was supported by ESPRIT Basic Research Action 6353 (NATURE) which is concerned with Novel Approaches to Theories Underlying Requirements Engineering and by the state Nordrhein-Westfalen, Germany.

1 Introduction

In software engineering, the requirements engineering (RE) phase has traditionally been perceived as the fuzzy and somewhat dirty part in which a formal specification is gained out of informal ideas. Everything prior to the formalization tends to be lost: in theory due to the inability to integrate informal information into a formal specification, in practise through burying requirements knowledge in bulky, practically unusable and unmaintainable documentation.

In contrast, research results show that requirements traceability is crucial for the success of system development activities (cf. [Dorfman and Thayer, 1990; David, 1990; Alford, 1990; Gotel and Finkelstein, 1993; Ramesh and Edwards, 1993]). As pointed out by [David, 1990] requirements traceability can be divided up into

- *pre- traceability*, dealing with the interrelation between the requirements and their origin (business goals, persons, user needs, etc.), thus allowing traceability from requirements back to their origin and vice versa;
- *post-traceability*, enabling the trace from each requirement down to design objects as well as implementation and documentation objects, but also from these objects, e.g., an implemented module, back to the requirements.

Two recent surveys support this distinction. Moreover, they indicate that pre-traceability is even more important than post-traceability (cf. [Gotel and Finkelstein, 1993; Ramesh and Edwards, 1993]). Despite this, most research has focused on requirements post-traceability so far.

In the following, we propose a framework for requirements pre-traceability which was developed within the Esprit project NATURE[2] (cf. section 2). Based on this framework we have developed a prototypical requirements engineering environment (PRO-ART). The architecture of the environment is described in section 3. Furthermore, each tool is briefly characterized in this section. Our planned demo scenario is depicted in section 4.

2 Brief Description of Research

What does it take to capture, maintain and use requirements information? The NATURE framework [Pohl, 1994] derives three dimensions (cf. figure 1) of interest from the major sources of problems in requirements engineering:

- **Representation Dimension.** As a basis for formal system development, requirements engineering moves along a representational axis typically from informal to more formal representations, ideally ending with a formal specification that can be transformed to executable code. Modeling and relating these representations is mostly a technical problem albeit one whose solution needs good knowledge of human-computer interaction.
- **Specification Dimension.** A second obvious goal is to end up with a complete system specification. We see this as orthogonal to the first problem: it is perfectly possible to hide poor understanding behind a lot of formalism, as well as to describe a very deep understanding in quite informal terms. Moves along this axis mostly face cognitive and psychological problems of requirements engineering.

[2] **N**ovel **A**pproaches to **T**heories **U**nderlying **R**equirements **E**ngineering (Esprit Basic Research Action No. 6353). A overview on the NATURE project can be found in [Jarke *et al.*, 1993a; Jarke *et al.*, 1993b]

- **Agreement Dimension.** The third axis is concerned with the agreement reached on the current specification. Requirements engineering is understood here as a negotiation process in which stakeholders coming from different context, e.g., customers or system managers, try to reach agreement on requirements, i.e. the specification. Hence this axis mainly deals with the social aspects of requirements engineering.

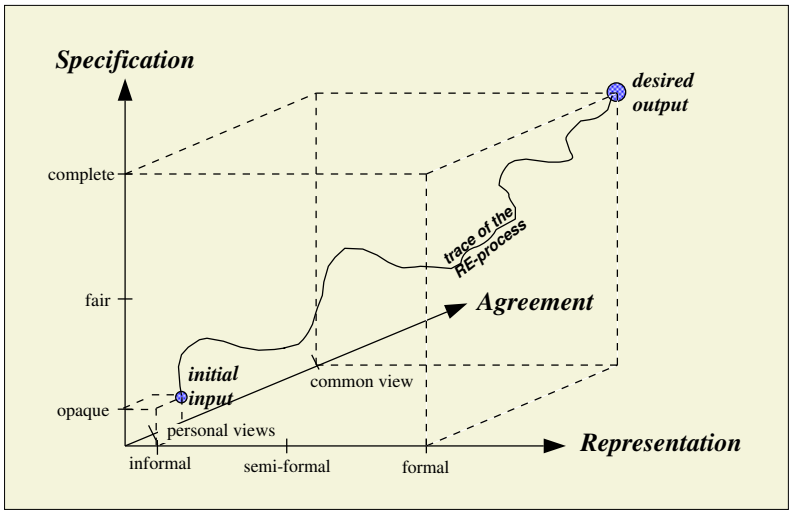

Figure 1 The RE process within the three dimensions.

At the level of process execution, RE can be understood as a movement in this space which, in addition to the technical, cognitive, and social barriers, is also influenced by economical factors and the methods used. For example, the traditional requirements capture process starts near the origin with different personal views, little system understanding, and a completely informal representation; it should end with agreement on a well-understood and formally described specification.

Meta-Modelling Approach: Based on the three dimensional framework, we have developed a set of traceability models to enable content oriented capture of the requirements engineering processes. These models include

- a hypertext model for recording informal information
- an entity relationship and structured analysis model for capturing semi formal information
- a comprehensive specification model gained out of over 20 standards and guidelines for structuring the primary product of the requirements engineering process; the specification
- an IBIS-based argumentation model for capturing the results of the negotiation processes, i.e. the decisions and their rationale
- a dependency model, providing different dependency links.

All these models were formalized using the knowledge representation language Telos [Mylopoulos *et al.*, 1990]. Interrelation between the objects are represented using dependency links, which are formally defined within the dependency model, or by multiple instantiation, e.g., the object *book* is an instance of *entity-type* (a class of the entity relationship model) and at the same time an instance of *data-store* (a class of the structured analysis model).

Capturing and interrelating all the trace information manually is almost impossible. Moreover, it would add a lot to the workload of the people performing the RE process. Therefore, the question *How to capture this information ?* must be answered.

Tool Interoperability Approach: Within the NATURE project we have developed a tool interoperability approach which is described in [Jarke *et al.*, 1994]. Briefly, our approach enables the cooperation of tools on the level of actions. Each action can communicate with other tools, respectively their actions. This is necessary, since actions performed during RE are often nested. The communication between the actions is controlled by a communication manager, thus enabling process guidance and control. Furthermore, the execution of actions is traced by recording (a) the products obtained, (b) the dependencies from existing products to old ones along any of the three dimensions, (c) the agent or tool actions creating these dependencies, plus some other annotations (e.g., criticality of the dependency). To achieve uniformity, decisions are treated like other objects when it comes to traceability, i.e., they depend on certain input objects and other objects or even the performance of actions may depend on them.

Both, our tool interoperability approach and meta-modelling approach build the basic for our prototypical requirements engineering environment described below.

3 PRO-ART: PROcess based Approach for Requirements Traceability

The current implementation of our prototypical requirements engineering environment (over 100.000 LOC) was implemented using C^{++} and the ANDREW tool kit, which is based on X11 R5. For representing the meta models for requirements traceability and the traces of process executions, we use the deductive object base ConceptBase [Jarke, 1991], an implementation of Telos [Mylopoulos *et al.*, 1990].

Figure 2 covers a coarse architecture of the PRO-ART prototype environment which currently consists of eight tools. Each of them is based on a formal tool meta-model and can work on a defined subset of the traceability models. The tools are briefly explained in the following.

Enabling Tool Interoperability

Communication Manager. To encapsulate tool interoperability properties and to insulate the individual tools from control knowledge about other tools, the NATURE environment offers a communication manager (CM) besides the process repository (cf. figure 2). Similar to Hewlett-Packard's SoftBench, the CM controls the communication among currently operational tool instances. That is, it knows (a) which tools are currently running, (b) how they can be accessed, and (c) what situation (mode and objects) they are in (i.e., whether they can provide the required services at present).

This approach has several advantages. Firstly, the CM hides details of message protocols from the individual tools. Secondly, as all tools are invoked (and tool modes changed) through messages, the CM can easily maintain the status of the process situation at any given point. Last not least, to support process adaptability, the CM also abstracts from the grouping of services into tools. If a tool needs a service performed, it needs not know which (combination of) tools offer this service because this information will be brokered by the CM. This is further strengthened by associating default tools for providing a service in particular types of process chunks. Taken

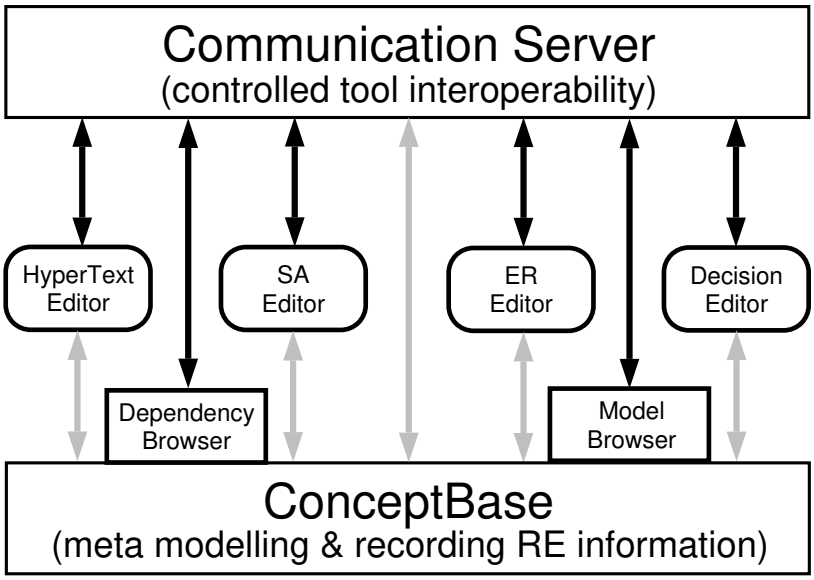

Figure 2 Architecture of the PRO-ART Environment

together, we have the following scenario. A service requests a sub-service from the CM. The CM finds the appropriate tool that offers this service, makes it available in the right mode and with the right objects, and restricts its use to the current process state. After the process steps associated to the sub-service have been executed, control is returned to the calling tool.

Capturing Knowledge along the Representation Dimension

In current RE praxis, specifications are defined using informal documents. Structured analysis and entity relationship modelling together with their graphical notations (DFD diagrams and ER diagrams) are quasi standard within industry. To demonstrate how this different representations could be integrated by tracing the RE process, we capture informal information within the hypertext editor and semi formal information by providing an SA and ER editor. Our overall approach is not limited to this representations and is currently being extended by object oriented analysis notations.

Hypertext-Editor. For recording informal information, we provide a new kind of hypertext editor. The editor allows the dynamical creation of hypertext nodes and their interrelation to all other objects, e.g., entities, during process execution. As all our tools, the hypertext editor enables traceability by recording the execution of each action in the central repository. Moreover, the hypertext editor provides services, which can be accessed via the communication manager. An example for such a service is the selection and creation of nodes. If this service is requested from an other tool, e.g., the ER-editor, the hypertext editor changes its modus and returns the created or/and selected objects back to the communication manager, i.e. the ER-editor (cf. figure 4).

Entity-Relationship-Editor. The entity relationship model, formally defined within Telos, is part of the requirements specification. Using the well known semi-formal graphical notations the entity relationship model can be graphically displayed and modified using the ER editor. Via the communication manager, the ER editor is able to access services from all other tools. For example, suppose that the creation of an entity depends on some informal stated requirements.

Theses informal requirements must be related to the new created entity to ensure traceability. During the execution of the *create entity* action the ER-Editor requests the service *select informal objects* from the communication manager. The communication manager translates this request into a corresponding service request for the hypertext editor (knowing that a hypertext editor is currently running). This request is sent to the hypertext editor, which first changes its modus (according to the current process status) and second allows the selection of parts of the informal requirements specification. The selected objects (hypertext nodes) are returned to the ER-editor (via communication manager) and are related to the just created entity using the dependency editor; again in a modus dependent on the process status (cf. figure 5).

SA-Editors. The structured analysis environment consists of two editors, one for creating and maintaining data flow diagrams and the other for the data dictionary. The DFD-Editor, as the hypertext and entity-relationship editor, provides services to the environment on the level of actions, e.g., the "highlight objects" action can be accessed from other tools. This services can be used, e.g, for emphasizing dependent objects (cf. figure 8).

Capturing the Knowledge along the Agreement Dimension

Decision-Editor. The decisions made during the requirements engineering process as well as their rationale are captured using the decision editor. As all other tools, the decision editor provides various tool modi and services, which can be used by other tools. An example of such a service, is the revision of a decision (cf. figure 6). Of course, also the decision editor records the execution of each action using the correspondent dependency link types.

Capturing the Knowledge along the Specification Dimension

Based on over 20 different standards and guidelines for requirements specification, e.g. IEEE-830, we have developed a comprehensive object oriented specification model. By classifying the informal requirements using this model, we are able to generate specification documents according to various standards, e.g., IEEE-830 or DOD-2716A, out of the current informal specification. Currently a tool is under development for supporting the instantiation of our model and for generating the different views.

Capturing Relationships

Dependency-Editor. This tool provides two essential services. First, the dependency stored within the knowledge base can be retrieved. The context of each dependency (i.e., their semantic) is captured by five attributes (cf. figure 7). Each of these attributes can be specialized or instantiated, to allow content oriented retrieval of dependency. The retrieved dependencies are displayed in a table (cf. figure 9). Second, the attributes for the dependency links created during process execution are specified using the dependency editor.

Dealing with the Traceability Model

Model-Browser. To deal with the complex traceability model and to allow comfortable retrieval of recorded objects, e.g., decisions, we have developed a model browser. Depending of the context reached during process execution, the model browser selects the suitable classes of the

meta model and displays it to the user. Furthermore, the user can browse through the instances of each class and can even retrieve the tool-categories, which can work on the selected instance. Moreover, by selecting a tool-category a concrete tool can be started out of the model browser, which will automatically load the selected object together with its context, e.g., the entity *book* and the corresponding ER diagram (cf. figure 3).

4 Overview of Demonstration

Within this section we give a brief overview of our planned demonstration. As demo scenario we have chosen the well known library example [Wing, 1990]. Our demonstration consists of three parts, each with different aims:

I. demonstrate the benefits of requirements traceability within the RE process
II. demonstrate the creation of dependencies between different objects (decisions, entities, hypertext nodes)
III. demonstrate the propagation of changes and their consequences to the specification using traceability information

Part I: Use of Traceability Information

Step 1: A new requirement is introduced into the informal specification *"Proceedings should only be checked out for three days (R1b)."* (cf. figure 4).

Step 2: Since there exists now entity *proceedings*, the entity *book* is identified as a possible generalization of proceedings (cf. figure 3).

Step 3: Looking at the history of the entity *book* (the decisions made, their rationale) a decision is retrieved (using dependency links) which says *"there should be no specialization of book"*. By retrieving the informal requirements which are related to the decision, additional information is gained:*"journals should not be checked out (R4)"* (cf. figure 5).

This information raises a new topic for the next requirements engineering meeting: the integration of the two requirements *R1b* and *R4*. Based on the retrieved information (the decision, the alternatives, the informal requirements etc.) a decision is made: integrate requirements R1b and R4 by specializing the entity *book*.

Part II: Recording process execution

Step 4: Back to his working place, the requirements engineer records the decision made during the team meeting. He revises the old decision *"there should be no specialization of book"* (cf. figure 6). During this activity, the informal requirements *R1b* and *R4* are related to the revised decision.

Step 5: Based on this decision, a specialization of the entity *book* into *proceedings* and *journals* is made together with the creation of the corresponding attributes. During the changes, dependencies between the different representations are created. Moreover, decisions (capturing the knowledge about the agreement dimension) are interrelated with the representations (informal, semi formal, formal) and linked to the specifications (cf. figure 7).

Step 6: The traceability of the changes made during the demonstration is illustrated by emphasizing the related objects of the just created entity *proceedings* (cf. figure 8) and the retrieval of all dependency links created during the demonstration (cf figure 9).

Part III: Integration of Changes

Step 7: Suppose, the requirement *"Proceedings should only be checked out for three days (R1b)."*, which was introduced in step 1, is revised after a few month. The dependency links enable the requirements engineer, to retrieve all objects which are related to the informal requirement (R1b). Hence, the recorded decision can be retrieved. Moreover, all actions which are based on this decision (with all the inputs and outputs produced) can be retrieved and displayed to the requirements engineer. In other words, the **relevant** parts of the recorded information (along the specification, representation and agreement dimension) can be retrieved, e.g., all the activities performed based on the informal requirement (R1b) or all the objects which depend on this informal requirement (cf figure 10).

Acknowledgments. We are grateful to our students Michael Gebhardt, Friedhelm Gibbels, Peter Haumer, Ralf Klamma, Christof Lenzen, Ute Loeb, Klaus Weidenhaupt and Claudia Welter who have implemented major parts of our prototype environment.

Bibliography

[Alford, 1990]
M. W. Alford. Software Requirements Engineering Methodology (SREM) at the Age of Eleven — Requirements Driven Design. In P. A. N. anbd R.T. Yeh (Ed.), *Modern Software Engineering*, pp. 351–376. IEEE, Van Nostrand Reinhold, 1990.

[David, 1990]
A. M. David. The Analysis and Specification of Systems and Software Requirements. In R. Thayer and M. Dorfman (Eds.), *Systems and Software Requirements Engineering*, pp. 119–134. IEEE Computer Society Press — Tutorial, 1990.

[Dorfman and Thayer, 1990]
M. Dorfman and R. H. Thayer. *Standards, Guidelines and Examples on System and Software Requirements Engineering*. IEEE Computer Society Press – Tutorial, 1990.

[Gotel and Finkelstein, 1993]
O. Gotel and A. Finkelstein. An Analysis of the Requirements Traceability Problem. Technischer Report, Imperial College, Department of Computing, TR-93–41, 1993.

[Jarke *et al.*, 1993a]
M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe and Y. Vassiliou. Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis. In *Proceedings of the 1th Int. Symposium of Requirements Engineering*, pp. 19–35, San Diego, CA, 1993. North Holland.

[Jarke *et al.*, 1993b]
M. Jarke, K. Pohl and S. J. et. al. Requirements Engineering: An Integrated View of Representation, Process and Domain. In *Proc. of 4th ECSE*, pp. 100–114, Garmisch-Partenkirchen, Germany, 1993. Springer Verlag.

[Jarke *et al.*, 1994]
M. Jarke, H. Nissen and K. Pohl. Tool Integration in Evolving Information Systems Environments. In *Workshop on Information Systems and Artificial Intelligence*. Springer Verlag, 1994.

[Jarke, 1991]
M. Jarke. (editor). ConceptBase V. 3.0 User manual. Technischer Report, Report MIP-9106, University of Passau, Germany, 1991.

[Mylopoulos *et al.*, 1990]
J. Mylopoulos, A. Borgida, M. Jarke and M. Koubarakis. Telos: Representing Knowledge about Information Systems. *Transactions on Information Systems 8*, 4 (1990), pp. 325–362.

[Pohl, 1994]
K. Pohl. The Three Dimension of Requirements Engineering: A Framework and Its Application. *Information Systems 2*, 19 (May 1994).

[Ramesh and Edwards, 1993]
B. Ramesh and M. Edwards. Issues in the Development of a Requirements Traceability Model. In *Proc. International Symposium on Requirements Engineering*, January 1993.

[Wing, 1990]
J. M. Wing. A Specifier's Introduction to Formal Methods. *Computer 23*, 9 (September 1990), pp. 8–24.

Figure 3 The model browser with a view on subset of the classes of our traceability model

Figure 4 The informal specification after adding the new requirement R1b.

Figure 5 Retrieving the history of the enity *book* (decisions and informal objects).

Figure 6 Revision of the decision *no specialization of books*.

Figure 7 Creation of dependencies links during the *specialize entity* action.

Figure 8 Emphasizing all objects which are related to the entity *book*.

Figure 9 The dependency editor displaying the dependencies created during the demonstration.

Figure 10 Retrieving the objects which depend on the informal requirement (R1b).