CREWS-99-04

Adapting Traceability Environments to Project-Specific Needs

Ralf Dömges, Klaus Pohl

Appeared in
Communications of the ACM, Vol. 41, No. 12, December 1998.

Adapting Traceability Environments to Project-Specific Needs*

Ralf Dömges and Klaus Pohl

Requirements traceability is defined as the ability to describe and follow the life of a requirement, in both a forward and backward direction. It implies that the life of each requirement can be understood from its origin, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration [6]. Requirements traceability is a prerequisite for effective system maintenance and consistent change integration [2].

Neglecting traceability or capturing insufficient and/or unstructured traces leads to a decrease of the system quality, causes rework and thus increases development costs and time. It results in a loss of knowledge if individual leave the project, leads to wrong decisions, misunderstanding and wrong communications [8,11].

Recent empirical research, such as described in the companion paper by B. Ramesh as well as in [6, 8, 11, 12], shows that systems management practice is progressing from the initial simple compliance verification schemata to very sophisticated models and policies for requirements traceability (see table 1 for examples).

Table 1: Example of traceability data and their expected benefits.

Traceability Data Types	Examples for Possible Benefits
bi-directional links [8,11], e.g. between customer requirements, derived requirements and software components	The interrelations support the (automated) 1. *validation* that the system functionality meets the customer requirements and that no superfluous functionality has been implemented. The links enable to check if a customer requirement is associated with a system component and vice versa. 2. *impact analysis* upon changing customer requirements. The links allow to determine which derived requirements, design and implementation parts might be affected.
contribution structures [6], e.g. stakeholders, roles, policies	Exploiting established contribution structures 1. improve *communication and cooperation* among teams. In the case of a change request, the stakeholders to be involved and/or informed can be determined. 2. assure that the *contributions* of (leaving) individuals *are passed on* and thus are not lost.
design rationale [5,7], e.g. design decisions, alternatives, underlying assumptions	Capturing design rationale improves 1. *change management* by reducing the chances of neglecting important considerations during change integration, since previously rejected solutions and the reasons for their rejection are accessible. 2. the *understanding* of the system by the customer and thus the system *acceptance*. The system behavior can be justified using the recorded design decisions and the recorded assumptions about expected operating conditions for the system.
process data [8,11], e.g. tasks performed, consumed resources, quality measures	Process data empowers the project manager to improve 1. *software project planning*, e.g. planning can be based on past histories of similar projects and thus results in more realistic cost and schedule estimates; 2. *software project control*, e.g. project progress can be measured based on the data collected through binary reporting and/or tracking techniques

* This work was supported in part by the DFG Project "Prozeßintegration von Modellierungsarbeitsplätzen", the DFG Collaborative Research Center 476 (IMPROVE), the Commission of the European Communities under ESPRIT Project 21.903 (CREWS), and by a cooperation grant from the German DAAD and the US National Science Foundation.

However, the same studies also point out that full capture of all conceivable traces according to these advanced models is neither desirable nor feasible under considerations of project cost and time. If requirements traceability is not "customized" it can lead to an unwieldy mass of unstructured and unusable data which will hardly ever be used [6, 9]. The adaptation of trace capture and usage to project-specific needs is thus a prerequisite for successfully establishing traceability within a project and for achieving a positive cost-benefit ratio.

The traces to be captured are influenced by factors like project schedule and project budget [11], by the contract, the development standards applied, existing laws, or the expected trace usage. A number of examples, drawn from focus group data of the study reported in the companion article by Ramesh, are sketched in the side bar „Examples of project-specific trace capture and usage".

Examples for project-specific trace capture and trace usage

The trace data to be recorded may depend on project-specific factors, e.g. contract obligations, laws, actual project phase, product part modified, individuals involved. To characterize the large variety we provide some brief examples:

Contract and project phases: Assume that the contract specifies periodical product reviews as well as the review procedure. To prove that the reviews were performed as defined, the products undergoing the review, the review results and the participants must be interrelated with the contract. The kind of interrelations, however, depends on the project phase. During the preliminary studies no interrelations are required by the contract. During the specification phase detailed interrelations have to be established; including the recording of design rationale. During the design and implementation phase, the products produced must be related to the specification, and thereby a indirect link to the contract is established. During maintenance and operation phase only internal reviews are performed. An relation to the contract is not required anymore. This specification by the contract clearly influences the type of trace data to be recorded during the different project phases. In addition to the contract project-specific guidelines define to consider review results before specification (parts) which were approved by the review are modified.

Ownership and privacy: Traces can be a valuable deposit of personal or organizational knowledge that needs to be protected. They uncover process know-how which may be the result of many years of experience. In many countries, labor laws or negotiated contracts pose limitations on what may be traced. Moreover, many developer companies want to make a careful distinction between what part of their traces constitutes documentation to be delivered to the customer (see above) and what constitutes knowledge capture for internal short-term or long-term purposes.

Process improvement: Assume that a certain project phase has produced a lot of defects. In future projects the steps performed in this phase may be captured in more detailed to provide the basis for their improvement.

People involved: An inexperienced project member might be asked to capture more detailed traces which provide the basis for approving her/his decisions by an expert. Moreover, inexperienced project members might be provided with much background information during process execution than the experts.

Standards and guidelines: For example, the British Standard BS 6719 concentrates on determining and documenting the requirements of (potential) users of off-the-shelf packages, bespoke systems and bureau services. BS 6719 demands recording data about the required system and its environment (e.g. the minimum expected life of the required system or the ability to recover from system failure). The American DoD-Standard 2167A focuses on the process of developing, testing and evaluating embedded real time systems. Consequently, it demands the recording of requirements, hardware and software items, risk assessment reports, test and evaluation procedures. To

The increasing use of commercial requirements traceability environments by industry reflects that requirements traceability is recognized as important success factor. A vendor questionnaire survey of commercial requirements traceability environments performed by INCOSE (INternational COuncil on Systems Engineering) studied the technical facilities of traceability environments, such as system environment, user interfaces, support and maintenance, and their capabilities for managing requirements, e.g. configuration management and trace analysis (see http://www.incose.org/workgrps/tools/tooltax.html). The survey included, among others, the following market-leading products: CORE® (Vitech Corporation), DOORS (Quality Systems & Software Limited), icCONCEPT RTM (Integrated Chipware, Inc.), RDD-100® (Ascent Logic Corporation), RequisitePro

(Rational Software Corporation), SLATE™ (TD Technologies, Inc.™), XTie-RT® (Teledyne Brown Engineering).

In this survey, vendors claim the following key capabilities of their environments:

1. pre-defined and customizable data types which can be captured manually or automatically by using data import facilities, parsing mechanisms and/or scripts;
2. pre-defined and user-definable queries for ad-hoc and customizable retrieval or grouping (e.g. filtering and sorting) of the data;
3. comprehensive configuration management and change tracking facilities;
4. trace analysis capabilities for identifying inconsistencies, performing impact analysis, verification, and (project) status accounting;
5. various presentation formats (e.g. matrices, graphs, reports) to display the data whose output can either be directed to the screen, to printing facilities, or different files formats which can be imported by publishing tools;
6. teamwork support by providing facilities to mutually notify stakeholders and for access control;
7. interfaces to other existing system and software engineering environment and tools as well as to publishing tools; further integration can be achieved by exploiting an applications program interface.

Clearly, some of these capabilities have been provided with the need for adaptability in mind. However, neither these environments nor the research literature offer a comprehensive concept of what project-specific adaptability requirements are, and how they can be provided in a coherent environment.

In the remainder of this article, we first elaborate on three key capabilities traceability environments need to offer to enable project-specific requirements traceability, and discuss to what degree they are supported by present environments:

- definition of project-specific trace data and tracing strategies;
- guiding stakeholders in project-specific trace capture and usage;
- empower organizational learning from captured traces.

Then, we present the design of a framework, called PRIME-RT, which offers a comprehensive solution to the identified shortcomings, sketch the implementation of two PRIME-RT-based, prototypical traceability environments and report on first experiences gained in traceability tasks in software engineering and chemical engineering.

Definition of project-specific trace data and trace strategies

The first obvious adaptability requirement calls for the ability of trace environments to define what traces are to be captured in what project situation. While present traceability environments do offer some flexibility in the definition of trace data types, they do not lend themselves easily to a situated approach to defining trace capture and usage strategies.

Definition of trace data types: To reduce the amount of recorded traces, the project manager has to specify the data types to be recorded within the project. Therefore s/he has to consider the expected trace usage, contractual obligations, laws and experiences of similar projects. This is an essential activity which has to ensure that the defined data types allow the recording of all required traceability information.

To empower the project-specific definition of data types, a traceability environment must provide capabilities to define new data types and to adjust predefined types according to project-specific needs. Moreover, it should systematically support the project manager in considering recorded experiences and the actual needs. The traceability data can be classified into four main categories [9]:

- product data subsumes the data types required to document the requirements of the system, e.g. the data types required to record UML models or textual requirements specifications,
- supplementary product data subsumes additional artifacts which provide explanations or justifications for the documented requirements, e.g. decisions, rationale, business goals.
- process observation data subsumes data about the process execution, e.g. activities performed, resources used, stakeholders involved.
- dependency data represents typed relations between the three data categories defined above, e.g. a versioning-relationships or a justification-relationship.

Current traceability environments allow the definition of *project-specific traceability data types*. Most environments provide a set of predefined data types from which the project manager can select the ones to be captured during the project. In addition, s/he can define new data types by either

- copying a predefined data type (e.g. CORE, icCONCEPT RTM), or
- subtyping a predefined data type (e.g. SLATE), or
- creating an instance of a generic meta data type (e.g., RDD-100).

Definition of trace strategies: In addition to the data types, the project manager must specify in which situation which data has to be captured, how the required data should be recorded and by whom. We call such definitions trace capture strategies. Moreover, the project manager may define possible sources for the traceability data and/or support for assessing the required information. Equally important is the definition of project-specific trace usage strategies. The project manager must define in which situations, which recorded traceability information has to be considered. The project manager may also define the use of aggregated, simplified or generalized information. The consideration of trace usage and capture strategies during process execution ensures correct trace recording and usage.

Some commercial environments provide script languages for strategy definition. For example, DOORS provides a c-like language DXL, icCONCEPT command-line macros, and SLATE exploits Tcl/Tk. These languages can be used to automate routine tasks, e.g. to calculate computed attribute values. Within a script, the recorded traces and most of the tool functionality can be accessed. Often, the scripts can be activated via menu options. Some environments also allow to associate scripts with data types; these scripts are executed whenever an instance of the associated data types is added or changed.

Still, the definitional capabilities of present environments leave a lot to be desired:

- *project-specific trace strategies are mainly defined the programming level.* There is no specification nor a design of project-specific trace strategies. In some environments (e.g. ARTS, CORE, SLATE, XTie-RT) even the project-specific data types are programmed. This makes a consistent definition, the adaptation and reuse very time consuming and extremely expensive, if not impossible, as experiences in software engineering indicates.
- *the definition of project-specific trace strategies and data is not guided.* Rather, the definitions rely on the experience of the project manager.

Guiding the stakeholder in trace capture and usage

Defining project-specific traceability definitions does not guarantee that the traces are correctly captured and used. To ensure definition-conform trace recording and usage, the definitions must be communicated to the stakeholders performing the process and must be applied in the corresponding situations throughout the project. Otherwise it is quite likely that the project-specific definitions are not adequately considered. Thus, defined data may not be recorded [7] and the recorded data may not be considered [5]. There are two principal solutions to achieve this: *handbooks* and *integrated method guidance* (see [8] for details).

Handbooks: The data types and strategies are defined in a project-specific traceability handbook. The usual way of orchestrating definition conform traceability is: during a learning phase each stakeholder learns the definitions. If the stakeholders do not remember the definitions in the corresponding process situations, trace capture and usage will not correspond with the definitions. In other words, the actors must know what are "legal" or "good" (method) steps according to the handbook and under which circumstances these steps can be applied. Of course the stakeholders can use the handbook as reference manual, especially if it is web-based, but they have to know and remember when to look for what.

Integrated method guidance: Ideally, the traceability environment guides (reminds, enforces, controls) the stakeholders in recording and using the traces according to the actual project-specific definitions. Assume that a project-specific definition specifies that for mission critical product parts the design rationale must be captured in an IBIS like structure, for less mission critical parts the rationale should be recorded as a textual statement and that for all other parts recording of design rationale is not required. Depending on the product being modified, the environment should request the recording of the defined rationale.

Similar, the interactive tools of the environment should remind the stakeholders to consider the recorded traces according to the definitions. Assume that whenever a stakeholder modifies an approved component, s/he has to consider the review comments and the approval conditions. Consequently, the environment should notify the engineer about the change conditions, retrieve the corresponding traceability data and display this data to the engineer.

A prerequisite for providing such definition-conform integrated guidance is that the interactive tools of the environment are able to interpret the project-specific definitions and are able to adapt their behavior accordingly. That means that the interactive tools must be process-integrated [10]. Offering integrated guidance to the engineer conformant with the project-specific definitions provides major advantages:

- The engineers must not necessarily be aware of the project-specific definition. They must thus not worry about the use or capture of the right traces in the actual situation;
- The training effort required is reduced;
- It ensures trace recording during process execution. Thereby trace re-construction is avoided. Reconstructed traces are typically incomplete and idealized in order to met certain expectations.
- It ensures that the recorded trace data is used. Thus, specifications of higher quality are developed;
- The workload of the stakeholders is reduced, since the traces can be recorded automatically whenever possible.

The guidance provided by existing environments is mainly twofold.

First, most environments restrict the data to be captured to data types specified for the project and support the correct instantiation of those data types.

Second, by defining and checking scripts consistent instantiation of different data types can be ensured (e.g. DOORS, SLATE). In other words, similar to triggers in active data bases, rules can be defined which reject inconsistent instantiations and notify the user about conflicts. Scripts can also be used to define the allowed modification of data. For example, the deletion of data on which other data depends can be denied. In practice, scripts are mainly used to ensure data consistency. However, the script languages can be used to implement trace capture and trace usage strategies. For example, environments which provide a comprehensive language like Tcl/Tk, enable the definition of additional interaction windows, the invocation of tool functions, the programming of complex dialogs etc. Theoretically, the script language can be used to implement a "new" tool.

Concerning the integration with other tools, the environments support data-integration by providing (standardized) import/export formats, e.g. CDIF. In contrast, control and process integration is almost not supported.

Still, present traceability environments remain too cumbersome because they suffer insufficient tool integration and do not offer integrated user guidance:

- *Weak integration with other engineering tools*: Current traceability environments are more or less seen as (data-integrated) "stand-alone" tools. Data integration does not guarantee that changes made to artifacts not known by the traceability environment are correctly reflected in the traceability data. Thus unintended inconsistencies can occur.
- *Insufficient activation of trace strategies (scripts)*. Scripts are either invoked by the user or invocation is based on events caused from data changes. The automated activation of scripts based on data changes is not sufficient. Rather, the activation needs to driven by the overall process and has to be based on the process definition, the actual process state, the process history and the goal a stakeholder tries to achieve (see [8] for details).
- *Script execution hardly influences the guidance provided*. The integration of user-defined scripts with the support offered by the traceability environment is, if at all, only coarse grained. Consequently, the script definitions can be in conflict with the traceability definitions, and the stakeholders can violate the script definitions using the functionality provided by the environment.

Empower organizational learning

In most organizations only limited knowledge about project-specific requirements traceability exists [6]. If available at all, the knowledge is not explicitly defined. There is thus no explicit process of defining high quality, project-specific trace definitions. The quality of the definitions almost entirely depends on the experiences of the stakeholders. As experiences in engineering [3] and other areas indicate only continuous process improvement leads to the production of higher quality products.

It is therefore essential to establish and continuously improve organizational knowledge about project-specific trace definitions. More precisely, the organization should learn when, for what purpose and how (using which strategy) a particular type of data should be captured and/or used. The accumulated experiences can than be used to guide a project manager in defining project-specific trace definitions.

To empower such organizational learning, the explicit definition of project-specific traceability strategies and a reflection of the experiences gained on the definition is required. To support the improvement, the traceability environment must facilitate the identification of strategy definition which have caused traces of poor quality or which led to insufficient trace consideration.

Existing environments record and maintain project traces in relational (e.g. ARTS, RequisitePro, ic-CONCEPT RTM) or object-oriented database systems (e.g. DOORS, RDD-100, SLATE). Most environments offer static and dynamic analysis mechanisms to check product inconsistencies and completeness. For example, it can be checked if all user requirements are satisfied by the defined system requirements and cyclic changes can be detected. Existing environments provide thus some means for analyzing the recorded traces. Defects detected can indicate a need for an improvement of the project-specific definitions. However, there remain a number of deficiencies which often prevent the proactive use of traceability data:

- *Lack of process orientation makes process improvement almost impossible*: Current traceability environments almost totally neglect the process responsible for the captured traces. The lack of appropriate interrelations between the traces and the steps performed makes a reflection of the experiences on the method definitions almost impossible. Consequently, insufficient product quality can only be detected, but not systematically avoided in future.

- *Organizational learning is not systematically supported*: Organizational knowledge is only captured at the project history level. No systematic procedure is established for guiding and controlling the learning from experiences. This shortcoming and, equally important, the lack of appropriate abstractions lead to a mainly personalized learning, i.e. learning is restricted to the experiences accumulated by the stakeholders.

PRIME-RT: A Framework for Project-Specific Traceability

We have developed an architectural framework for adaptable, project-specific traceability environments which offers solutions to the shortcomings identified in the last section. Figure 1 presents an overview of our architecture which consists of three main components: a *method definition*, a *process execution*, and a *method improvement environment*.
Note that each part of our architecture adapts well-known recent techniques from literature or advanced practice. This does not just give our approach some empirical face validity but also enables partial uptake of the most important improvements by vendor and user organizations.

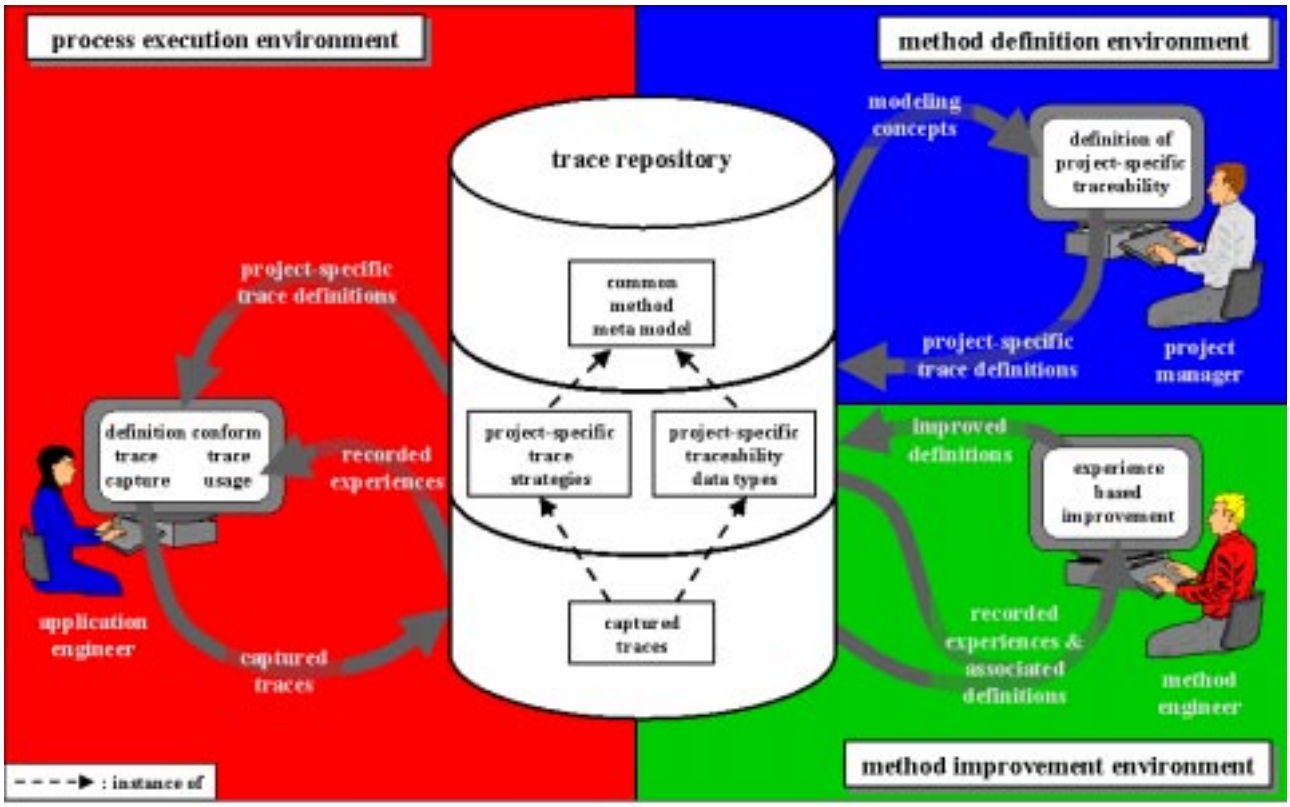

Figure 1: An architecture for adaptable traceability environments

The Method Definition Environment: Defining Project-Specific Traceability

To offer guidance for a project manager, traceability environments should adapt the underlying ideas of method definition environments which support the incremental, model-based definition of project-specific methods [1].

Guiding the definition of methods: Method engineering approaches empower the construction of a method by selecting, combining and adapting a set of reusable, model-based *method fragment*s. A method fragment defines both, the methodical advice and the data to be produced and considered. The definition and selection of appropriate fragments according to project-specific needs is supported, e.g. by contingency factors which characterize the support offered by each fragment ac-

cording to a given need. Recording the fragments in a repository provides the foundation for selective retrieval and thus facilitates their reuse.

Model-based method definition: Briefly, method engineering approaches provide modeling languages to define method fragments on a more abstract requirements/design level. This is a prerequisite to systematically support the project manager in reusing and adapting existing definitions according to project-specific needs. Compared with a direct implementation of method fragments a model-based definition leads to increasingly consistent definitions, and facilitates the correction of errors, as well as the maintenance of project-specific definitions.

We have adapted these capabilities for our method definition environment. Thereby the definition of project-specific traceability data and trace strategies and their integration into the overall RE process was improved.

The Process Execution Environment: Process-Integrated Trace Guidance

To achieve the demanded definition conform trace capture and usage within the process execution environment we have adapted the main ideas underlying process-centered environments (PCEs, [4]) and process-integrated environments (PIEs, [10]). The improvements achieved are:

Definition conform process execution: A PCE supports the process execution based on the interpretation of explicit method definitions. It consist of a set of interactive tools used to perform the process. The tools can exchange data, e.g. via a trace repository. Moreover, they can share their services. The process engine, a specialized tool, coordinates the interoperation of the tools according to an explicit method definition and the actual process state. Thereby a definition conform process execution is guaranteed. Achieving project-specific trace guidance requires however an integration of the trace strategies with the method definitions.

Integrated method guidance: Process-integrated environments go one step further and provide process-integrated tools which offer integrated method guidance to the stakeholders. In a PIE the interactive tools are able to interpret the method definition. In addition, the process engine passes the actual process state to the tools which, in turn, adapt their user interfaces accordingly. Briefly, based on the interpretation of the method definitions and under consideration of the actual process state, the tools adapt the guidance offered by restricting the products and the menu options displayed. The integrated and adaptable guidance guarantees the consideration of the actual traceability definitions during process execution (see [10] for details).

Process-driven strategy activation: In contrast to existing traceability environments, our architecture enables data-driven, user-driven and process-driven activation of the defined strategies. A strategy can be activated via user-interactions, by the trace repository (e.g. if an integrity constraint is violated) and by the process engine (e.g. during the execution of a strategy). It is the integration of trace strategies and method definition which establishes a process-driven activation of the trace strategies. This kind of activation considers, in addition to the recorded data, the actual process situation and the method definitions.

Interrelating captured traces with strategy definitions: Similar to commercial traceability environments, the tools and the process engine of a PIE record the traces in a trace repository. In addition, they relate the traceability data with the project-specific strategy definitions which have caused their creation or usage. They thereby establish an important prerequisite for experience based process improvement.

The Method Improvement Environment: Supporting Organizational Learning

To empower organizational learning an incorporation of process improvement approaches is required. Improvement approaches like CMM or TQM have proven successful in many application areas. Those approaches differ (slightly) in the way they incorporate people, but they are based on the same principle. Striving for continuous process improvement requires four major prerequisites: (1)

explicit definition of the method; (2) guide process performance based on the method definitions; (3) record experiences about process execution; i.e. process execution must be traceable; (4) relate the captured traces to the method definitions.
The technical implications of those requirements are quite straight forward. First, the trace strategies must be explicitly defined and persistently recorded. Second, the process execution environment must guide the stakeholders based on the actual definitions. Third, the process execution environment must record the traces and relate them with the strategy definitions. Fourth, the method definition environment must support the analysis and generalization of the captured experiences. As described above our architecture fulfills these four requirements and thereby establishes the basis for organizational learning.

Experiences

Based on the PRIME-RT framework we have implemented two prototypical *adaptable traceability environments* [9,10]: PRO-ART supporting project-specific traceability in requirements engineering processes and TECHMOD supporting project-specific traceability during the modeling of complex chemical processes. Both environments provide a process execution and a method definition environment. The method improvement environments are currently under investigations.

The *process execution environments* consist of a process engine, and various, process-integrated editing and browsing tools (e.g. an OMT-editor to define requirements, a flowsheet editor to define chemical process models, a decision editor to capture design rationale). The process engine controls and monitors the tools according to the project-specific definitions. Our *method definition environment* enables the model-based definition, selective-retrieval and the adaptation of project-specific definitions. The traces and the trace definitions are persistently stored and interrelated in a RDBMS-based *trace repository*.

We applied both prototypes in experiments and small case studies. Briefly, the experiences confirmed that the project-specific trace guidance leads to better trace capture and trace consideration and thus to higher product quality. Without the integrated method advice, important traceability data would have often not be adequately recorded. It turned out, that the method definition environment supported the project manager in reusing project-specific trace definitions for different experiments even within different application domains. Moreover, in the case of method deficiencies the definitions were easy to adapt. Both benefits facilitated and accelerated the method definition. The experiences further indicate that in most cases, the recorded traces and (maybe more important) the consideration of the recorded traces was conform with the project-specific definitions.

Besides this positive feedback, also some shortcomings have been identified: The adaptation of trace strategies according to project-specific needs could be improved by providing a framework which classifies the strategies according to expected trace usage and trace obligations and by providing trace filters which ease the project-specific adaptations. Trace usage requires appropriate presentation and aggregation formats. The question which format should be used in which situation to present a certain type of data is an open research issue.

References

[1] Brinkkemper, S., Lyytinen, K., and Welke, R.J., Eds. Method Engineering: Principles of construction and tool support. Chapman & Hall, London, England, 1996.

[2] Conklin, J. Design Rationale and Maintainability. In Proceedings of the 22nd International Conference on System Sciences (January , Hawaii, USA) 1989.

[3] Deming, W.E. Out of the Crisis. Massachusetts Institute of Technology, Center for Advanced Engineering Study, Cambridge, 1986.

[4] Finkelstein, A. C.W., Kramer, J., and Nuseibeh, B., Eds. Software Process Modelling and Technology. Advanced Software Development Series, J.Wiley & Sons Ltd., Taunton, England, 1994.

[5] Fischer, G., Lemke, A.C., McCall, R., and Morch, A.I. Making Argumentation Serve Design. In *Design Rationale: Concepts, Techniques, and Use*. 1996. pp. 267-293.

[6] Gotel, O.C.Z. Contribution Structures for Requirements Engineering. PhD Thesis, Imperial College of Science, Technology, and Medicine, London, England. 1996.

[7] Gruber, T.R. and Russell, D.M. Generative Design Rationale: Beyond the Record and Replay Paradigm. In *Design Rationale: Concepts, Techniques, and Use*. 1996. pp. 323-349.

[8] Pohl, K. Process Centered Requirements Engineering. Advanced Software Development Series, J.Wiley & Sons Ltd., Taunton, England, 1996.

[9] Pohl, K., Dömges, R., and Jarke, M. Towards Method-Driven Trace Capture. In Proceedings of the 9th International Conference on Advanced Information Systems Engineering (June, Barcelona, Spain). 1997.

[10] Pohl, K. and Weidenhaupt, K. A Contextual Approach for Process-Integrated Tools. In Proceedings of the 6th European Software Engineering Conference (Sept., Zürich, Switzerland). 1997.

[11] Ramesh, B., Stubbs, C., Powers, T., and Edwards, M. Implementing Requirements Traceability: A Case Study. Annals of Software Engineering 3 (1997), 397-415.

[12] Stehle, G. Requirements Traceability for Real-Time Systems. In Proceedings of the EuroCASE II (April, London, England). 1990.

Ralf Dömges (doemges@i5.informatik.rwth-aachen.de) is a doctoral candidate at Informatik V, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany.

Dr. Klaus Pohl (pohl@i5.informatik.rwth-aachen.de) is a senior Research Associate at Informatik V, RWTH Aachen, Ahornstr. 55, 52074 Aachen, Germany.