Decision Oriented Process Modelling

Klaus Pohl, Ralf Dömges, Matthias Jarke
Informatik V, RWTH Aachen
Ahornstr. 55, 52056 Aachen, Germany
{pohl, doemges, jarke}@informatik.rwth-aachen.de

Abstract

We propose decision-oriented process modelling as a step towards human-centered process management, and demonstrate some implications of this model for the interaction between process modelling, process enactment, and process performance in a CASE environment. We also discuss the potential our approach offers for experience-based process improvement.

1 Introduction

The treatment of humans within a **P**rocess **C**entered (software) **E**ngineering **E**nvironment (PCEE) is still an open question. At the 8th ISPW, a differentiation between the static approach (humans are treated as subroutines that are invoked by the environment) and the dynamic approach (the environment provides a virtual machine upon which the human executes some portions of the process models) was made [8]. [13] suggested to treat the users dependent on the role they play in the software engineering process. Human involvement in software processes was defined by nine dimensions [12]. Nevertheless, all these contributions neglect one important aspect, namely **decisions** and their rationale made during process execution.

There are many reasons for integrating decisions into process definitions [16]. Our approach was mainly influenced by two main reasons.

Application engineers react contextually according to the domain knowledge they acquire, by analogy with previous situations they have been involved in. In order to take this into account, the engineer must be supported in making decisions. Process guidance must not only provide the definition of actions to be performed next. In addition, arguments for and against performing an activity (decisions) and situations in which the activities can be performed (the decision context) have to be integrated in a process model [5].

In most existing approaches, the reasons why a certain action should be performed in a given situation is hidden to the humans performing the process. This neglects an important prerequisite for process improvement, namely the ability to understand the process definitions (the reasons behind them) and therefore to enable criticism of the definitions based on the reflection of the current experience. Process improvement (in the spirit of TQM) requires that people performing the process feel responsible for producing higher quality, which itself requires at least a sufficient understanding of process definitions.

In the following we briefly describe the decision oriented meta model for supporting design processes developed and implemented within the NATURE[1] project (section 2). A concrete way of working, called *guidance model* is represented as instantiation of the process meta model. Due to decisions made by the application engineers, process execution may differ from process definition. To record process deviations and thereby enable traceability of process performance, a second structure, called *traceability structure* (cf. section 3), is needed. Our **R**epository based **A**pproach for **P**rocess **C**entered **E**ngineering **E**nvironments (RAPCEE) offers an integration of the process model and trace structure, enables argumentative process guidance, and automates the recording of process traces (section 4). Moreover, interrelating recorded traces and process models enables experience based process improvement.

2 A decision oriented meta model

The decision oriented meta model developed in NATURE is an extension of approaches developed in the ALECSI [14] and the DAIDA [2] projects. DAIDA introduced a model of goal-oriented and argumentation-based decision-making, while ALECSI emphasized the situated nature of process actions. The five basic concepts of our

[1] Esprit Basic Research Action 6353 NATURE (**N**ovel **A**pproaches to **T**heories **U**nderlying **R**equirements **E**ngineering)

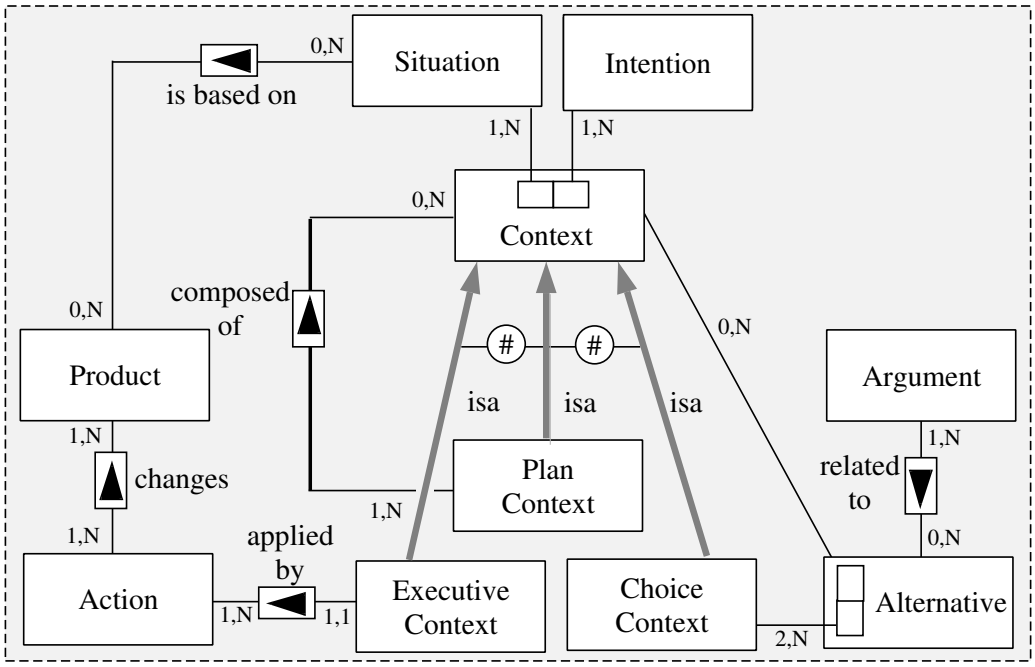

Figure 1: The NATURE decision oriented process meta model [5, 6]

model are therefore the ones of *situation, intention, context, action and argument.*

Figure 1 introduces the key concepts of the basic meta model and their relationships with a binary entity-relationship notation. A *situation* is most often a part of the product it makes sense to make a decision on. Situations are thus built from product parts of the product undergoing the development process. An *intention* reflects the goal to be achieved in a given situation. A *context* is composed of a situation and the intention applicable in the given situation. The decision-oriented approach implies three kinds of contexts.

The part of the process definition, in which the application engineer has to make decisions is called *choice contexts*. Within a choice context, at least two *alternatives* exist. An alternative can be another choice context, a plan context, or an executive context. The application engineer can choose one of the alternatives, or even introduce a new one. *Arguments* (pros and cons) are related to the alternatives and guide the application engineer in making a decision. By choosing one of the alternatives, the engineer refines the intention. Thus, the right part of the figure 1 can be referred to as goal refinement.

A strategy to fulfill a certain intention is represented as a *plan context*. A plan context represents a certain order on a subset of arbitrary contexts. It can introduce new choice, executive, and other plan contexts. A plan context can be supported by forcing the application engineer to deal with the contexts in the order defined by the plan. Since a plan describes which sub-goals (*contexts*) must be fulfilled to reach the initial goal, this cycle can be referred to as goal decomposition.

The *executive context* leads immediately to the performance of an action. It represents the part of the process definition which can be strictly enforced, or even automated. The application engineer does not have any choice what to do next. An executive context is operationalized by performing the *action* related to this context. Performing an action changes the product and may generate new situations, thereby activating new contexts. Thus the left part of figure 1 can be called product transformation or goal satisfaction (cf. [14, 15, 5] for details).

So far, we have described each of the three kinds of contexts separately. Additionally, we have to consider the hierarchical structure of plan and choice contexts embedded in each other. This structure can be seen as an AND/OR graph where the plan contexts are seen as AND nodes and the choice contexts as OR nodes. Mylopoulos et al. [7] have pointed out that AND/OR graphs are useful for organizing process knowledge about the satisfaction of nonfunctional requirements and goal-oriented requirements engineering; more generally, this hierarchy can be developed dynamically to mix process planning with process execution — the typical pattern for creative processes like requirements engineering.

Summarizing, the NATURE process meta model offers an infrastructure for what could be called *multi-perspective process models*, i.e. for the integration of at

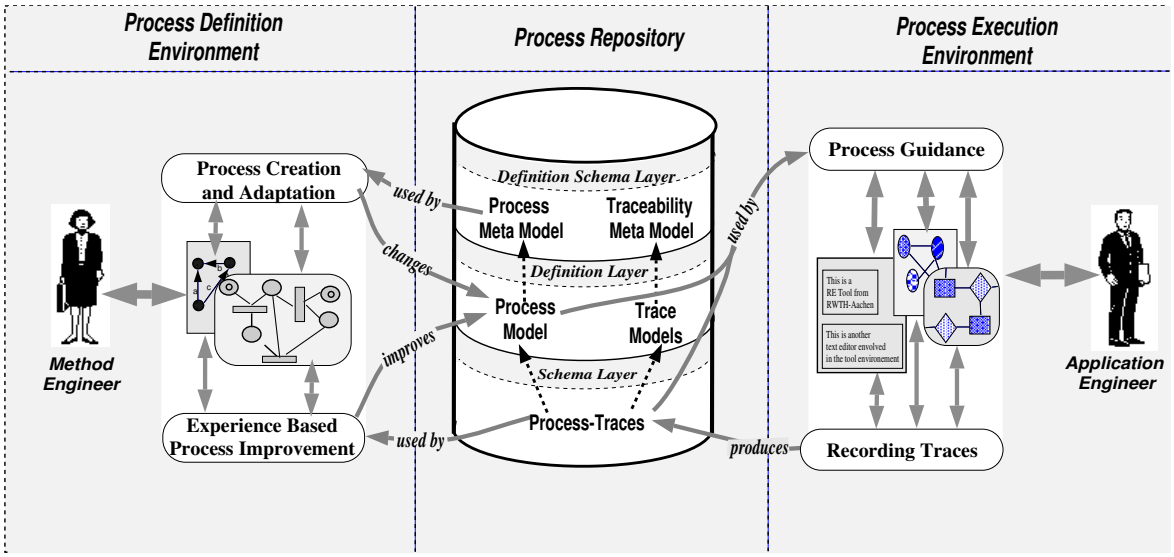

Figure 2: RAPCEE–Repository based Approach for Process Centered Engineering Environments.

least four different process modeling paradigms in a very sparse conceptual framework.

3 Process guidance and trace

A specific *way-of-working* (the process model) is defined using the decision-oriented meta model introduced above. As usual, the process model is used for process enactment.

In contrast to most existing approaches, the meta model implies that process execution can be either

- automated, whenever an executive context is active
- enforced, whenever a plan context is active or
- guided, whenever a choice context is active.

The alternatives applicable in a choice context are ranked. Moreover, arguments are provided to the application engineer for supporting his decision making process. Pro and contra arguments for choosing a particular alternative are captured by the process model and displayed to the application engineer. It is up to the application engineer to choose one of the alternatives, based on the defined arguments and his particular experience. He may even introduce a new context which was not considered as a possible alternative by the process model. Therefore, process execution may differ from the process definition. In order to make the process execution traceable and to enable experience-based process improvement, deviations made during process execution must be captured.

Hence, two conceptually different kinds of models are needed: descriptive and prescriptive ones. Descriptive *traceability models* offer a design record structure by which the process trace is to be organized. Prescriptive *process models* define the intended way-of-working in the process.

However, the two structures must at least make the actual and the intended trace comparable. Traceability and process models are therefore organized according to process and traceability meta models which are specializations of our decision oriented meta model shown in figure 1. The main additions needed for the process meta model are granularity abstractions and control structures for organizing the process, whereas the main additional concepts of the traceability meta model is that of processors (human agents and tools) who execute a process in a specific context at a specific time.

A prescriptive process definition is then a instance of the process meta model which we call process chunk. Similarly, the traceability meta model can be instantiated to trace model chunks, i.e. methods to organize a descriptive recording of process executions.

4 Repository based PCEE

We have developed a **R**epository based **A**pproach for **P**rocess **C**entered **E**ngineering **E**nvironments, called **RAPCEE** which enables the different kinds of process support (guidance, control, enforcement), the capturing of process traces, as well as the improvement of process definitions based on the trace information. As depicted in figure 2, RAPCEE consists of three parts, the *process definition environment*, the *process repository*, and the *process execution environment*, which are briefly characterized below.

Process repository (center of figure 2): The various kinds of data are recorded and interrelated in a central

repository, which is based on the **I**nformation **R**esource **D**ictionary **S**ystem (IRDS) framework standard (cf. [1]). At the upper level (*Definition Schema Layer*), the languages for representing concrete process models (*Process Meta Model*) and the traceability models (*Traceability Meta Model*) respectively, are defined. These languages are specializations of the common meta model introduced in section 2 (not shown in figure 2). Whereas these languages are domain independent, the process models and the trace models defined at the middle level (*Definition Layer*) are domain dependent. For the domain of requirements engineering, we have defined a traceability structure according to the three dimensions of requirements engineering (cf. [9]). Finally, at the lower level (*Schema Layer*), the process traces are recorded as instances of the process definitions and trace models.

Process execution environment (right side of figure 2): Process enforcement, guidance, and control is based on the defined *way-of-working* (the process model) and integrated into the engineering environment used by the application engineer (upper right of figure 2). The support offered depends on the kind of process chunk which is currently active, i.e. either an automated action is performed (*executive context*), or the application engineer is forced to follow a certain plan (*plan context*), or the possible alternatives, together with argumentation structures, are shown to the application engineer who has to make a decision (*choice context*). A pre-defined process chunk can be activated either by the application engineer, e.g. by selecting an alternative, or by the process engine interpreting the process model, e.g. activating a new context according to an active plan context. Thus, the process performance domain as well as the process enactment domain can both activate a process chunk. This is in contrast to most existing approaches, where only the process enactment domain (process engine) has access to the process definitions and enforces process performance.

Traceability is an important issue in system development for many reasons: product change and reuse support; identification of mission-critical components; recording of design rationale; project tracking and cost estimates etc. The process execution environment must ensure that the process trace is recorded according to the defined traceability structure. In addition, the trace must be related to the process definition to enable experience-based process improvement. For example, if the application engineer has chosen an alternative within a choice context based on arguments not captured in the process definition, these arguments must be recorded and related to the decision they belong to. Only in this case can the decisions be re-constructed and used for process improvement. Furthermore, there should be a graceful degradation of the trace, when application engineers do not follow the defined process model. Exceptions and aberrations must be expressible and rationalizable. Generalization hierarchies of product and process models provide a basis for the graceful degradation of information.

Summarizing, the execution environment must provide process support based on the current process situation (dependent on the active process chunk) and record the trace information to enable traceability as well as experience based process improvement.

Process definition environment (left side of figure 2): There are two main activities to be supported by the process definition environment:

(1) defining concrete ways of working (process model) according to the process definition language;
(2) improving these models according to experiences made during process execution.

For the first kind many possible solutions have been proposed in the literature, e.g., specialization hierarchies. However, supporting the process improvement activity is largely neglected, except for modelling the improvement processes themselves. RAPCEE enables the support of this activity based on the process traces. Moreover, since process traces and process definitions are interrelated, the method engineer is able to retrieve the process chunks which lead to a particular trace.

However, improving the current way-of-working based on experience captured by the trace should not be ad-hoc. Using our decision oriented meta model, improvement activities can be modeled like normal process chunks. These process chunks are modeled at the same level as the process models (at the *Definition Layer*). The interrelation of process models and traces allows us to make improvement more precise. If process execution differs from the definition, for instance because

- an action not covered by the process model was performed,
- a new argument was introduced during a decision,
- a decision not covered by the process model was made,

the corresponding process improvement chunks are instantiated by the process execution environment during process performance and the recorded improvement chunk is marked as an open task for the method engineers.

5 Conclusions

Based on the framework sketched above, we have implemented a prototypical environment, called **PRO-ART** (**P**rocess and **R**ep**O**sitory based **A**pproach to **R**equirements **T**raceability, cf. [3, 10, 11] for details). PRO-ART enables different kinds of process support and experience based process improvement based on the content oriented recording of process executions. So far, pro-

totypes of the main architectural mechanisms (repository models, communication server, guidance and traceability interfaces) and of several prototypical process integrated tools (ER, SA, hypertext, decision editor) have been implemented with a total of more than 100.000 lines of source code.

In contrast to most existing approaches, process performance is either automated (when an executive context is active), enforced (when a plan context is active), or guided (when a choice context is active). Furthermore, a process chunk can be activated either directly by the application engineer (by selecting an alternative) or by the process engine (enforcing a plan context or executing the action related to an executive context). Since the alternative selected by the application engineer could be a plan context, the selection can lead to the enactment of a new plan context, i.e. to the execution of a new process engine.

Within the NATURE project (cf. [4]) the implementation is being used as a platform for evaluating and possibly improving many different kinds of requirements engineering techniques (methods).

Acknowledgments

This work was supported by the ESPRIT Basic Research Project 6353 NATURE (**N**ovel **A**pproaches to **T**heories **U**nderlying **R**equirements **E**ngineering) and the ISI Project ECAUS003 (**I**nformation **S**ystems **I**nteroperability). We are grateful to our NATURE partners for many discussions and contributions, especially the group of Colette Rolland at Paris-Sorbonne, for their contributions to the process model. Major portions of the PRO–ART prototype were implemented by our students Michael Gebhardt, Friedhelm Gibbels, Peter Haumer, Ralf Klamma, Christof Lenzen, Klaus Weidenhaupt, and Claudia Welter.

References

[1] ISO/IEC. *Information Technology – Information Resource Dictionary Systems (IRDS) – Framework*, 1990. ISO/IEC International Standard – 10027.

[2] Matthias Jarke. *Database Application Engineering with DAIDA*. Springer Verlag Berlin Heidelberg, 1993.

[3] Matthias Jarke, Hans Nissen, and Klaus Pohl. Tool Integration in Evolving Information Systems Environment. In *3rd GI Workshop Information Systems and Artificial Intelligence: Administration and Processing of Complex Structures*, Hamburg, Germany, 1994.

[4] Matthias Jarke, Klaus Pohl, Stephan Jacobs, Janis Bubenko, Petia Assenova, Peter Holm, Benkt Wangler, Colette Rolland, Veronique Plihon, Jean-Roch Schmitt, Alistaire G. Sutcliffe, Sara Jones, Neil A.M. Maiden, David Till, Yanis Vassilou, Panos Constantopoulos, and Giwrgos Spanoudakis. Requirements Engineering - An Integrated View of Representation, Process, and Domain. In *Proc. of the 4th Europ. Software Engineering Conf.*, Garmisch-Partenkirchen, Germany, 1993. Springer-Verlag.

[5] Matthias Jarke, Klaus Pohl, Colette Rolland, and Jean-Roch Schmitt. Experience-Based Method Evaluation and Improvement: A Process Modelling Approach. In *Proc. of the IFIP Working Group 8.1: CRIS '94*, Maastricht, Netherlands, 1994. Springer-Verlag.

[6] Mario Moreno, Colette Rolland, and Carine Souveyet. A Generic Approach to Support a Way-of-Working Definition. In *Proc. of the 6th Intl. Conf. on Advanced Information Systems Engineering*, Utrecht, Netherlands, 1994. Springer-Verlag.

[7] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.

[8] Dewayne E. Perry. Humans in the Process: Architectural Implications. In *Proc. of the 8th Intl. Software Process Workshop*, pages 124–125, Wadern, Germany, March 1993.

[9] Klaus Pohl. The Three Dimensions of Requirements Engineering: A Framework and its Applications. *Information Systems*, 19(3):275–292, 1994.

[10] Klaus Pohl, Petia Assenova, Ralf Dömges, Paul Johannesson, Neil Maiden, Veronique Plihon, Jean-Roch Schmitt, and Giwrgos Spanoudakis. Applying AI Techniques to Requirements Engineering: The NATURE Prototype. Sorrento, Italy, May 1994.

[11] Klaus Pohl, Ralf Dömges, and Matthias Jarke. PRO-ART: Process Based Approach for Enabling Requirements Traceability. Technical Report NATURE-Report No. 94-7-1, RWTH-Aachen, Germany, 1994.

[12] Samuel T. Redwine. Human and Process. In *Proc. of the 8th Intl. Software Process Workshop*, pages 12–14, Wadern, Germany, March 1993.

[13] Samuel T. Redwine. Supporting the Other Kinds of Software Process Users. In *Proc. of the 8th Intl. Software Process Workshop*, pages 129–131, Wadern, Germany, March 1993.

[14] Colette Rolland and Corine Cauvet. ALECSI - An Expert System for Requirements Engineering. In *Proc. 3rd Intl. Conf. Advanced Information Systems Engineering*, Trondheim, Norway, 1991. Springer-Verlag.

[15] Colette Rolland and Naveen Prakash. Reusable Process Chunks. In *Proc. of the Intl. Conf. Database and Expert Systems Application*, Prague, 1993.

[16] Thomas Rose and Matthias Jarke. A Decision-Based Configuration Process Model. In *Proc. of the 12th Intl. Conf. on Software Engineering*, Nice, France, 1990.