Knowledge Elicitation for Descriptive Software Process Modeling

Ulrike Becker-Kornstaedt
Fraunhofer IESE
Sauerwiesen 6
D-67661 Kaiserslautern
Germany
Phone: +49 (0)6301 707 135
email: becker@iese.fhg.de

Abstract

Descriptive software process modeling is a key activity in process engineering. However, a systematic approach on how to collect the information needed to develop such a model is needed. This position paper points out the major problems when doing descriptive software process modeling in industrial contexts and suggests to use qualitative research methods to develop a method for systematic software process elicitation.

Keywords:

Descriptive software process modeling, software process elicitation, software process modeling methodology, qualitative research techniques

1 Introduction

Descriptive software process modeling, that is, getting a description of the software process as it is being performed, is a key activity in process engineering. Little research has been done on how to systematically collect the knowledge needed to develop these models. This paper suggests to explore qualitative research methods for their application in the elicitation of knowledge needed to provide descriptive software process models. As only a subset of techniques can possibly be examined, the selection of potential candidate techniques should base upon models of ‘knowledge’ used psychology. Not all possible techniques are applicable in the industrial context in which descriptive software process modeling usually takes place, further delineating the range of possible techniques.

This paper is structured as follows: The next section defines the general problem of how process modeling is done and outlines the research done in the area. Section 3 describes the constraints a process engineer may meet when doing process elicitation in industrial practice. Section 4 describes an approach to software process elicitation and gives a rough background on qualitative research methods. Section 5 gives a summary and conclusion.

2 Problem:

Descriptive software process modeling is concerned with developing a description of the actual development process. Descriptive software process models are a key component in software engineering, e.g., to guide a process or as a basis for measurement in the context of software process improvement programs.

A major quality criteria for a descriptive software process model is its accuracy (i.e., the model has to reflect the actual process), so that any process-related activities can be based on the process as it is carried out. A process engineer who uses a non-accurate process model, for instance, as a basis for measurement program, may try to measure activities which are only depicted in the model but do not take place in reality. Figure 1 explains the role of descriptive software process modeling in process improvement and the major steps in descriptive software process modeling.

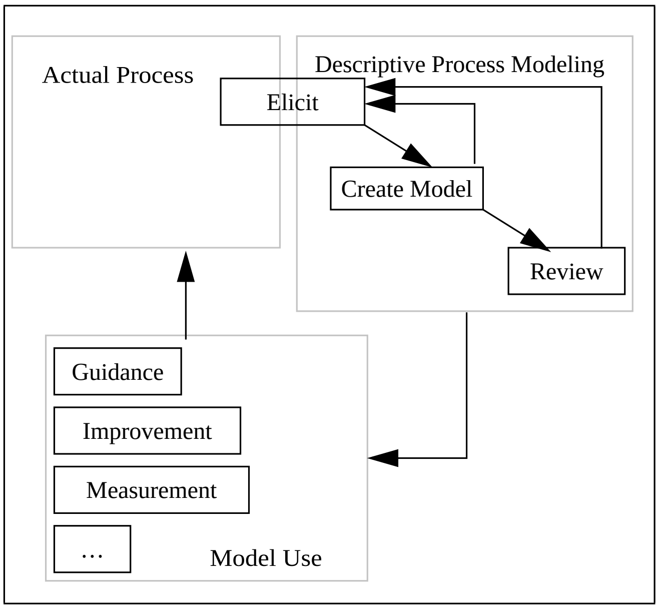

Figure1: Steps in descriptive software process modeling

Typical steps in descriptive software process modeling are elicitation of process knowledge, creation of models, and review of models by process performers [3]. Usually, there are rework cycles going from model creation back to elicitation, for instance, when important information is missing, and from review to the elicitation step, if it shows that the model is incomplete or incorrect. Thus, the elicitation step is a key activity in descriptive process modeling, as the other activities depend on it: The 'better' the outcome of process elicitation, the less rework is necessary.

Software process modeling is a starting point for many other activities, and the quality of the resulting process model has an impact on all other activities. Thus, process modeling, and should be made efficient and repeatable.

A lot of research has been done on *what* needs to be described in a process model, i.e., the contents (e.g., [2] and [13] propose schemas describing the relevant entities and relationships between these to be represented in a process model). Research has also been conducted on *how to formalize* the information contained in process models, i.e., the notation (for instance [10] compare different notations for process modeling against each other), or on the general approach of how to develop descriptive software process models (see for example [11]). Typical information sources for software process knowledge reported in the literature are interviews with process performers, analysis of process artifacts, and observation of process performers (see for instance [1], [4], or [8]). However, these describe where to get the information needed to obtain a descriptive software process model, but do not provide details on how to get the knowledge from these sources. [7] propose to use data collected on the events of an executed process to discover the behavioral aspects of the process. However, this method assumes that people already collect data on the process and know what data to collect.

Thus, a systematic approach to software process elicitation is needed.

3 Constraints in Practice

When exploring potential techniques for their use in software process elicitation, the following limitations which occur in industrial practice, have to be taken into account:

- Limited availability of process engineers [5], [3]: Process modeling in an industrial environment often requires that process modeling activities interfere as little as possible with everyday work of process performers. Even worse, often, process performers are not available to process engineers at all, or they may be available only for a limited amount of time.

  Thus, the direct interactions with them should be as short as possible and there should be as few direct interactions as possible.
- Geographical constraints: The organization where process modeling is being done and the organization that does process modeling may be in distant locations [5]. This implies that there are only few direct encounters on site, but many indirect encounters, for instance via phone, video conference, or e-mail.

  Thus, techniques for indirect interactions must be also taken into account.
- The actual process may not be the same process as the official one, i.e., the one described in the process handbook, and the understanding of the official process may vary among process performers [3].

  Thus, a strategy for software process elicitation can not solely rely on the official process handbook.
- Confidentiality of information: A lot of the information needed to develop the process model is confidential. This regards company-specific information (e.g., certain development practices, or specifics of the product under development) and information provided by process performers (e.g., a process performer may reveal that the actual process varies considerably from the official one, in a situation where an organization may be required by contract to follow a given process). It should be ensured that this information can be treated confidentially. Sometimes special techniques or question wording may have to be used to elicit this type of knowledge.
- Process performers may leave out aspects of the process [9]. Often process engineers may not be aware that they get incomplete information. Thus, it is very important that process engineers have some background information so that they can assess the quality and completeness of the information provided by process performers.
- People may have difficulties articulating what they did in concrete terms that can be mapped onto the abstractions the outside process engineers could express [6].

In addition to the problems directly related to process elicitation, there are some problems which are inherent to software development processes:

- Especially in large development processes, there are many different roles and individuals involved. The knowledge is scattered [50]. As not everybody involved in a process can be interviewed, sampling strategies for interviews are needed.
- Distributed locations: especially with large teams, different parts of the software process may be performed in different locations. This makes it more difficult to get a consistent and complete picture of the development process.
- Some process steps are performed very seldomly [5]. This may imply that process performers tend to forget these steps or are not able to report them accurately.

4 Approach

The purpose of process elicitation is to come up with data which allows an accurate description of the software process. The information sources available to a Process Engineer are documents (organizational documents, such as project plans as well as artifacts which are deliverables or intermediate results), interviews with process performers, and observation of Process Performers. Knowledge elicitation from artifact analysis, interviews, or observation has been done in other disciplines.

A field which has been dealing with these types of issues is *qualitative research methods*. Qualitative research methods refer to research procedures which produce descriptive data: people's own written or spoken words and observable behavior [12].

These techniques have been applied, refined, and adapted to match specific problems for a long time. A lot of qualitative research methods have been applied in non-technical areas, for instance Requirements Elicitation use qualitative research techniques. Thus, there exists not only a wide range of techniques themselves but also a lot of experience on how these techniques have been adapted, as well as on their applicability.

I propose to explore qualitative research techniques for their use in the elicitation of software process knowledge. Qualitative techniques are applied tor instance in ethnology to study populations or in psychology and sociology to find out about people's feeling.

In detail, the following issues are of interest for software process elicitation:

- Related to *artifacts*:
  - What artifacts can give a fast overview of the process?
  - What artifacts are useful to obtain a detailed knowledge of process steps?
  - What information can they provide (e.g., detail, what phases of the process)?
  - What are other characteristics to look for (e.g., structure, relationship to other documents)?
  - What techniques used in qualitative research can be used to exploit software process artifacts?
- Related to *interviews*:
  - What techniques/question types are adequate to get an overview of the process?
  - Structured interviews allow detailed information to be obtained rather efficiently. What structured interview techniques are suitable for process elicitation?
  - In focused interviews the interview focuses around a certain topic, e.g., certain documents. What process documents are useful to focus an interview on?
  - Sampling of interviewees: who/what role can provide what information, at what level of detail?
  - What interview techniques are suited to obtain detailed descriptions?
  - How can interviews be designed to be efficient (e.g., interview templates)?
- Related to *observation*:

- What information can be retrieved using observation?
- What observation techniques are adequate to provide the information needed?
- Observation in industrial contexts can not be done unobtrusively. What techniques can ensure valid data nevertheless?

A lot of process knowledge exists only implicitly in the minds of process performers. Here, it may be helpful to have a detailed look at what ‘process knowledge’ means from the (cognitive) psychology viewpoint. How can ‘process knowledge’ be related to mental models? This insight is sure to help to further select the most promising techniques.

As software projects differ a lot from each other, such an approach can not be a ‘one-fits-all’ method, but rather a series of techniques based on techniques used in qualitative research methods, together with guidelines when to use certain techniques, what the risks are, what pitfalls exist, and what the results are.

5 Summary and Conclusion

This paper describes the necessity for a systematic approach to software process elicitation to develop descriptive software process models.

Research has been done on what to describe and how to describe process models. Also there exists some experience report on where to obtain the software process knowledge from, but there is no systematic approach describing how to obtain the knowledge. Typical problems a process engineer has to face when doing process elicitation in an industrial environment are related to the limited availability of process experts, and to the lack of background knowledge on the process engineers’ side. Qualitative research methods are proposed as the basis of a systematic approach to software process elicitation.

References

[1] Jim Arlow, Sergio Bandinelli, Wolfgang Emmerich, and Luigi Lavazza. A fine-grained process modelling experiment at british airways. *Software Process–Improvement and Practice*, 3(3):105–131, November 1997.

[2] James W. Armitage and Marc I. Kellner. A conceptual schema for process definitions and models. In Dewayne E. Perry, editor, *Proceedings of the Third International Conference on the Software Process*, pages 153–165. IEEE Computer Society Press, October 1994.

[3] Sergio Bandinelli, Alfonso Fuggetta, Luigi Lavazza, Maurizio Loi, and Gian Pietro Picco. Modeling and improving an industrial software process. *IEEE Transactions on Software Engineering*, 21(5):440–454, May 1995.

[4] Naser S. Barghouti, David S. Rosenblum, David G. Belanger, and Christopher Alliegro. Two case studies in modeling real, corporate processes. *Software Process–Improvement and Practice*, 1(1):17–32, August 1995.

[5] Ulrike Becker-Kornstaedt and Wolfgang Belau. Descriptive Process Modeling in an Industrial Environment: Experience and Guidelines. In *EWSPT 7*.

[6] Brendan G. Cain and James O. Coplien. A Role-Based Empirical Process Modeling Environment. In *Proceedings of the Second International Conference on the Software Process*, pages 125–133. IEEE Computer Society Press, February 1993.

[7] J.E. Cook and A.L. Wolf. Automating process discovery through event-data analysis. In *Proceedings of the Seventeenth International Conference on Software Engineering*, pages 73 – 82.

Association of Computing Machinery, April 1995.

[8] John Favaro. Process modelling at the european space agency. In J. C. Derniame, editor, *Proceedings of the Second European Workshop on Software Process Technology*, Lecture Notes in Computer Science Nr. 635, pages 160–162. Springer–Verlag, September 1992.

[9] Marc I. Kellner and Gregory A. Hansen. Software process modeling: A case study. In *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, volume II, pages 175–188, 1989.

[10] Marc I. Kellner and H. Dieter Rombach. Session summary: Comparisons of software process descriptions. In Takuya Katayama, editor, *Proceedings of the Sixth International Software Process Workshop*, pages 7–18. IEEE Press, October 1990.

[11] Nazim Madhavji, Volker Gruhn, Wolfgang Deiters, and Wilhelm Schäfer. PRISM = methodology + process-oriented environment. In *Proceedings of the Twelfth International Conference on Software Engineering*, pages 277–288, 1990.

[12] Steven J. Taylor and Robert Bogdan. *Introduction to Qualitative Research Methods: A Guidebook and Resource, Third Edition*. John Wiley and Sons, 3 edition, 1998.

[13] Richard Webby and Ulrike Becker. Towards a Logical Schema Integrating Software Process Modeling and Software Measurement. In Rachel Harrison, editor, *Proceedings of the Nineteenth International Conference on Software Engineering Workshop: Process Modelling and Empirical Studies of Software Evaluation*, pages 84–88, Boston, USA, May 1997.

Ulrike Becker-Kornstaedt received the B.S. degree (Vordiplom) in computer science and the M.S. degree (Diplom) in computer science from the University of Kaiserslautern, Germany. She is a researcher within the Process Engineering and Technology group of the Fraunhofer Institute for Experimental Software Engineering in Kaiserslautern, Germany. Her research interests include descriptive software process modeling, knowledge elicitation and qualitative research techniques and their application to software engineering.