Formalizing the Software Process in Small Companies

Pablo Ruiz[1], Alcides Quispe[2],
María Cecilia Bastarrica[2] and Julio Ariel Hurtado Alegría[1],[2]

[1] IDIS Research Group, University of Cauca, Colombia
[2] Computer Science Department, Universidad de Chile, Chile

Abstract. Counting on a formally defined software process brings a series of benefits such as being able to improve project management, reusing the knowledge acquired about software development, comparing software work products across projects on a concrete basis, and making easier and less error prone the evolution of the software process itself, among others. However, building a software process is never easy, let alone formalizing it. In this paper we propose a meta-process, i.e., a process for formalizing the software process using the EPF platform. We show how the meta-process was applied in a medium size Chilean software company and we analyze its advantages and drawbacks.

1 Introduction

The benefits of counting on a rigorously defined software process as a basis for improvement both from the point of view of product quality and process productivity are usually agreed upon. Among these benefit we find: having a concrete basis for process analysis, evaluation and improvement, and a source of information for training new developers. Software process explicit definition enables certification or evaluation according to standards such as CMMI [19] or ISO [20] that may also bring commercial benefits.

However, it is usually considered not only costly but also hard to define, formalize, enact and institutionalize the process; these costs could be unaffordable for small companies [7,8,9]. Formalization requires specialized skills about software processes, mastering special notations and tools, and counting on the ever scarce time from developers and managers. Moreover, once having a defined process it is not clear that it would be useful for all projects, e.g., a large new development project would probably require a much more complex process than a simple and short maintenance project [13].

Some research has presented proposals attempting to help software companies when trying to formalize their software process [2][3][10][11][15]. Unfortunately these studies do not mention the size of the participating companies, or when it is mentioned, they are usually big companies. Therefore it is not possible to identify the studies results for small companies. For example, Becker-Kornstaedt and Belau [2] reported the experience of capturing the development process in a subdivision of the Space Infrastructure Division of DaimlerChrysler; however such division has about 100 employees only for software development.

In this paper we propose a meta-process, i.e., a process for defining and formalizing software processes. This meta process has been designed based on practical experiences in Chilean companies [16]. Our proposal is specifically defined for small companies so it is intended to be light-weight, it includes only a couple of easy to follow steps, it uses a standard notation for the formalization as SPEM 2.0 [17], and it suggests the use of EPF[3], an open source platform as the supporting tool.

We report the application of the proposed meta process in a small Chilean company. We started by trying to formalize the process they have already informally defined but soon we realized that the process actually applied differed dramatically from what was prescribed. Provided that no real improvement can be achieved on a fictitious process [6], we decided to elicit the actual process. The formalization followed is the proposed meta process. We found that only a few people realized the benefits of having a formalized process and thus their collaboration was not very enthusiastic. On the other hand, we counted on the support of the company's CEO and that resulted fundamental for the success of the formalization process. However, the company's process engineer in charge of maintaining the formalized process did not master the specification tool, so the adoption of the formalized process was not as successful as originally thought.

The rest of the paper is organized as follows. In Sect. 2 a series of related work is discussed. The proposed meta process is presented in Sect. 3. Section 4 describes the application of the meta process in the case of the pilot company. Finally, Sect. 5 draws some conclusions and delineates possible future work.

2 Related work

Modeling the actual software process is a key activity within process engineering. According to Becker-Kornstaedt [1], there is little research about systematically collecting information needed for defining these models. Normally models are defined by the process engineer using documents, interviews and observation of process performers. Becker-Kornstaedt [1] defines an approach to elicit process requirements using a qualitative research method. This method identifies a set of issues to elicit process requirements from each source of information: artifacts, interviews and observation. These issues are used for identifying key sources, relevant information and suitable techniques. However, this approach does not define a method or technique to specify the software process model; we use a formal language as SPEM 2.0 and a platform as EPF for process formalization.

A systematic, view-based approach to eliciting process models is proposed by Turgeon et. al. [21]. This approach includes underlying techniques and a supporting tool called V-elicit. Software process elicitation involves: gathering process information from agents involved in a development process, from documents, and

[3] Eclipse Process Framework Componer - `http://www.eclipse.org/epf/`

through observation; modeling this information; and verifying that the model built is consistent and complete. Process information is gathered from different sources, and separate descriptions called views are merged to form the entire model. The techniques underlying this approach include the following activities: planning for elicitation, eliciting the different views, checking for intra-view consistency, identifying common components across views, merging the views, checking for the overall model quality and modifying the model if necessary. The underlying techniques, together with the supporting tool, constitute a novel contribution in the software process field. In our approach we follow some of the steps suggested in this one.

Elicit [11] is a method to elicit processes from environments using three dimensions: views, methods and tools. The view dimension includes five elicitation perspectives (both static and dynamic): process steps, artifacts, roles, resources and constraints. The method and tool dimensions respectively define the elicitation steps and the support applied to any perspective. In Elicit the method dimension defines the following steps: understand the organizational environment, define elicitation objectives, plan the elicitation strategy, develop and analyze process models, usage post analysis and experience packing. We also take some of these steps for defining our meta process.

Colombo et al. [3] and Cook et al. [5] present data analysis techniques to discover software processes from enactment data. Under these techniques, data describing process events are first captured from an on-going process and then used to generate a formal model of that process's behavior. Hindle [10] proposes a software process recovery method using analysis of version control systems, bug trackers and mailing list archives with a variety of supervised and unsupervised techniques from machine learning, topic analysis, natural language processing and statistics. They combine these methods to recover process events that are mapped back to software development processes like the Unified Process producing diagrams called Recovered Unified Process Views (RUPV) that are similar to those diagrams used in the Unified Process.

Jacchery et al. [15] present E^3, a specific process modeling language and a supporting tool especially defined for process model elicitation. E^3 is an object-oriented modeling language with a graphical notation where associations are a means to express constraints and facilitate reuse. The E^3p-draw tool supports the creation and management of E^3 models and provides a view mechanism that enables inspection of models according to different perspectives.

Conradi et. al. [4] propose that formalizing a software process includes the specification of the development process itself as well as the process for evolving this process. Our meta process complements this conception by including the first step of initially formalizing the development process.

Our proposed meta process is light weight in the sense that it includes only a couple of simple steps. We also use SPEM 2.0 that is the OMG standards for process specification, as the formalization language, and EPF an open source platform. All these characteristics make our proposal appealing for small software companies.

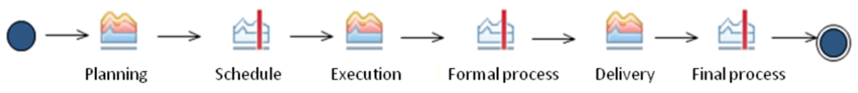

Fig. 1. Meta-process for process model formalization

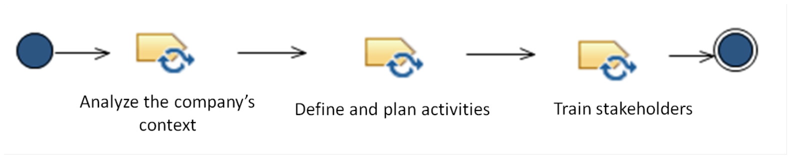

Fig. 2. Planning phase

3 Software meta process

We propose a meta process for formalizing software processes. It is composed by three main phases: *Planning*, *Execution* and *Delivery*, as shown in Fig. 1.

The output of the *Planning* phase is the definition of the activity schedule to be followed along the whole meta-process. The *Execution* phase is an iterative cycle for each process area, and its main milestone is the formal description of the whole process model that is also validated by all the stakeholders. The milestone in the *Delivery* phase is the final process formalized in the chosen language and available for enactment.

3.1 Planning phase

In this phase the first contact with stakeholders takes place and the planning itself happens. The process for this phase is divided into three main tasks: *Analyze the company's context*, *Define and plan activities*, and *Train stakeholders*, as shown in Fig. 2.

The goal of *Analyze the company's context* is to get involved with the characteristics of the company and the way things happen in it. To this end it is necessary to acquire information about the company's business goals and also about the process usually followed for developing software even though it may be neither defined nor formalized.

During the *Define and plan activities* task the activity plan is conceived during a meeting. All members of the formalization team including all company's stakeholders must take part in this meeting. It is of the utmost importance that the company's CEO attend this meeting as well so that the formalization scope could be determined and a person could be designed as the company's process owner. It is desirable to count on all the people in charge of each area for this meeting provided that they are the owners of the software process followed in their respective areas. The output of this task is the schedule where the meetings with the people responsible for each area in the company are defined.

During the *Train stakeholders* phase it is necessary to make all people involved with the software process familiar with the formalization language and the tool before starting the meta-process execution. In this way they could also be involved in the formalization process and they could actively participate in the elicitation and validation meetings. Moreover, this training is also useful for future process maintenance and evolution because these activities will not be necessarily the responsibility of a unique process engineer, but the process could be maintained by people in each area.

3.2 Execution phase

This phase is an iterative activity that is executed for each of the process areas in the company. It is composed by four tasks: *Obtain information*, *Analysis and Design*, *Specification*, *Partial Validation* and *Complete Validation*, as shown in Fig. 3.

The goal of *Obtain Information* is to elicit information about the software process that is actually applied in practice. Information such as: tasks, work products, tools, roles, guidelines and templates are elicited during this phase. It is necessary to interview the people in charge of each area and also their key collaborators if found useful. The purpose of this phase is to acquire the information necessary for the process analysis and design, that is the following stage.

During *Analysis and Design*, the formalization team studies and creates a draft of the flow of elements that are involved in the process. As a result of this task a first version of the general process should be obtained where all identified process elements must appear.

The goal of the *Specification* task is to take the draft software process and to formalize it using a specification language such as the SPEM 2.0. The result is a prototype of the formalized process that would be later validated.

During the *Partial Validation*, the prototype process specification is shown to the stakeholder in order to receive feedback. It must be investigated if the prototype captures the process that is actually applied in the company. For this task, the formalization team must meet the head of each area and his/her collaborators that had already taken part in the elicitation phase in order to show them the formalized process and to obtain their opinion. Suggestions and changes are applied to the formalized partial process.

For the *Complete Validation* a meeting takes place among stakeholders from all areas involved in the development process in order to visualize and discuss the complete formalized process. The goal of this task is to receive a variety of opinions that will help in fixing and optimizing the process as a whole.

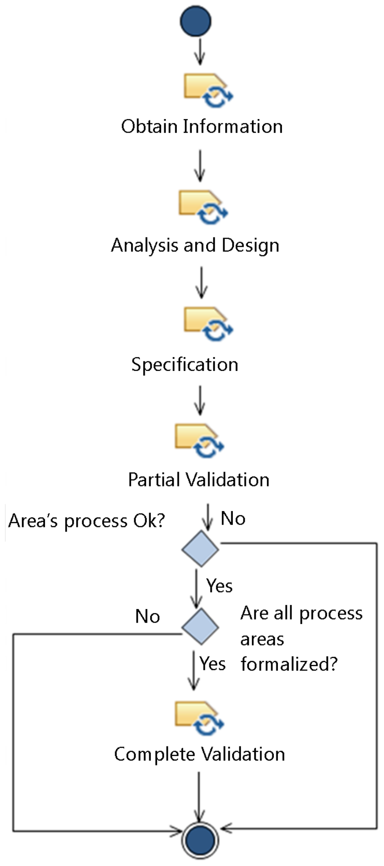

Fig. 3. Execution phase

3.3 Delivery phase

In this phase the result of the formalization is shown and delivered, as shown in Fig. 4. The final presentation of the software processes takes place so that it is available for usage. The specification of the software process formalized in the chosen specification language is delivered.

4 Application

The software process formalization meta process was applied in a small size Chilean software company. Since its creation this company has developed software products and services for the retail business. Its main product is a Point Of Sale (POS) software for the IBM 4690 system. The company was created approximately 20 years ago and it has grown slowly. Currently it also develops software for Peruvian and Colombian customers as well.

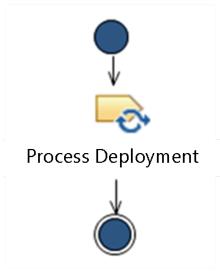

Fig. 4. Delivery phase

This company has recently decided to formalize its software development process by applying our proposed meta-process. Several working sessions, between July and August 2011 were conducted by two process engineers in order to formalize the company's software process. At the end of these sessions, the company counted with its software process completely modeled in the Eclipse Process Framework Composer tool.

4.1 Applying the Planning phase

The first step in the formalization of the software process was to get a preliminary understanding of the process context of the company (*Analyze the company's context*). The process engineers have investigated about the company and the software process currently applied. The web page of the company was used as a source of information about the company.

In order to have a preliminary understanding about the company's current software process, all the available documentation about its software process was requested. Two documents were obtained: one describing in detail the current software process (where the process was specified as a UML activity diagram) and another one with additional information about some new inputs and outputs, roles and descriptions to better clarify the process. After analyzing these documents it was clear that the software process followed by the company was based on the Unified Process approach (inception, elaboration, implementation and transition).

Next, a session for setting the stage (*Define and plan activities*) was scheduled. For this activity, it was requested the participation of the company's CEO and the head of each of the different development areas. The goals for working together with these people were: a) to give a clear explanation of the benefits of having a formalized software process; b) to clarify the scope of the current formalization of the software process; c) to identify the main phases of the software process in the company; d) to designate one process representative for each phase, in general the most experimented person, who would be in charge of defining the workflow with the process engineers; e) to schedule an elicitation session with each of the process representatives; and finally, f) to designate, among the participants, the representative of the company to act as the link between the process engineers and the company.

Having done the previous session, then another session was scheduled in order to train all process representatives in the SPEM 2.0 standard and the EPFC tool (*Train stakeholders*). In this session the process engineers briefly explained some important topics of SPEM 2.0 and developed some examples using the EPFC tool. The goal was to give the participants a concrete introduction about the platform that would be used in the formalization of the process in order to make future communication easier between the process engineers and the company process representatives.

4.2 Applying the Execution phase

Next, it was the time to develop the workflow for each phase of the software process. To achieve this goal the following activities were done: *Obtain information*, *Analysis and Design*, *Specification* and *Partial Validation*. These activities were performed looking to develop one workflow at a time.

Each elicitation session (*Obtain information*) was attended by: a) the process engineers; b) the representative of the company; and c) the process representative in turn (according to the schedule). The objective of these sessions was to develop each workflow including its activities and the roles and input and output work products involved in each activity. During each session one of the process engineers played the role of a moderator and the other one recorded all the discussion. The moderator drew the workflow on a blackboard while the process representative was interviewed by the two process engineers. Questions were focused mainly on getting information about the activities (including the roles and work products) usually performed by the company in this particular phase. At the end of the session we had a first full version of the workflow (including activities, input/output work products and roles) on the blackboard. The recorder engineer had also written in paper the entire workflow and all the relevant information discussed during the session. In general the resulting workflow was different from the one described in the process documentation that was given to the process engineers. However, the purpose of this phase was to document the process actually applied in each area.

Having written in paper the entire workflow, the next step was to model it into the EPFC tool. One of the process engineers was in charge of doing this task. He processed the information that was in paper to analyze it. Sometimes some activities were merged or deleted according to the notes that were taken in the elicitation session. Then the workflow was adjusted and was ready to be specified in the EPFC tool (*Analysis and Design*). Next, the workflow (activities, roles and work products) was specified in the EPFC and a HTML version of the process was generated and given to the other process engineer to make an internal validation. This validation was done against the workflow written in paper (*Specification*).

Figure 5 shows the specification of the *Implementation* process area in the company. The workflow depicts the first version developed by working together

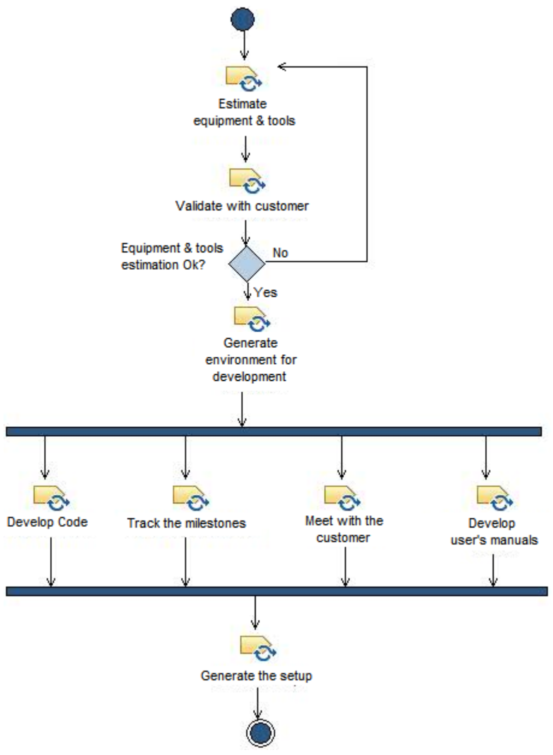

Fig. 5. First specification of the *Implementation* process area

with an experimented project manager of the company. According to him, before starting the code writing for any project at hand, it is necessary to have installed the required hardware and software infrastructure (*Estimate equipment and tools*). That is why the workflow starts by estimating such infrastructure and then sends it to the customer to get his/her approval (*Validate with customer*). Then it proceeds to install the environment (hardware and software) to start the software construction. Four main activities are performed in parallel during software construction: *Develop code*, *Track the milestones*, periodically *Meet with the customer* and *Develop user's manuals*. Once having a stable release version of the software product, the next step is to *Generate the setup* version, i.e., the version to be installed in the customer hardware infrastructure.

Next, the following step was to validate the EPFC version of the workflow with the process representative to whom it belongs (*Partial Validation*). The process representative navigated the HTML version of the workflow looking at the activities, roles and input/output work products in order to validate its completeness and correctness. Then, based on his/her feedback, the workflow was adjusted and validated again until having a final approved version. Figure 6 shows the adjusted version of the *Implementation* process area specification. For

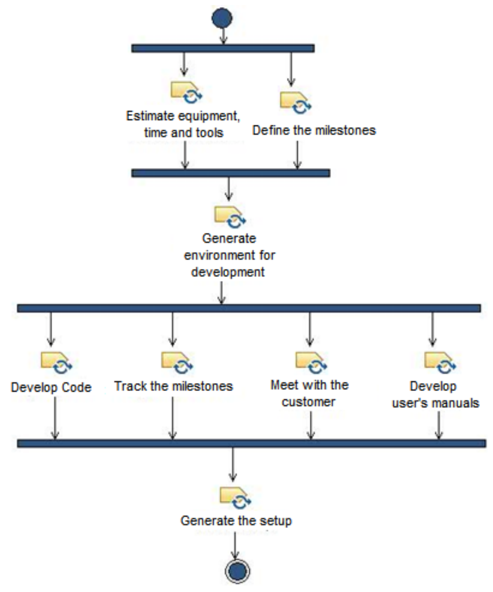

Fig. 6. Specification of the *Implementation* process area after *Partial Validation*

the *Partial Validation* session, the project manager came with a collaborator. They both proceeded to review the workflow modeled in the EPFC. Comments and observation of the latter were those that guided the adjustments to the workflow. At the end, the original workflow suffered three main modifications: the task *Estimate equipments and tools* became *Estimate equipments, time and tools*, the task *Validate with customer* was deleted, and a new task *Define the milestones*, in parallel to *Estimate equipments, time and tools*, was added. Then it was the time to fill the detailed textual descriptions of the tasks within each activity, the roles and the input/output work products of the workflow. The process representative filled this information in a given template and sent it by e-mail to the process engineers. This activity was done as many times as process areas identified as part of the company's software process.

Once all workflows were modeled in the EPFC, the next step was to schedule a session to validate them as a whole product (*Complete Validation*). For this session it was requested the participation of the process representatives who took part in the elicitation sessions. Again, in this session one of the process engineers played the role of a moderator and the other one as a recorder. During the session, the moderator showed the HTML version of the process in a data projector one workflow at a time. For each workflow the process representatives were free to make their observations. The recorder engineer was in charge of writing the observations in a hard copy of the process. Although every workflow was previously validated and approved independently, several observations rose during this session. It seemed that one person (no matter if he/she is an exper-

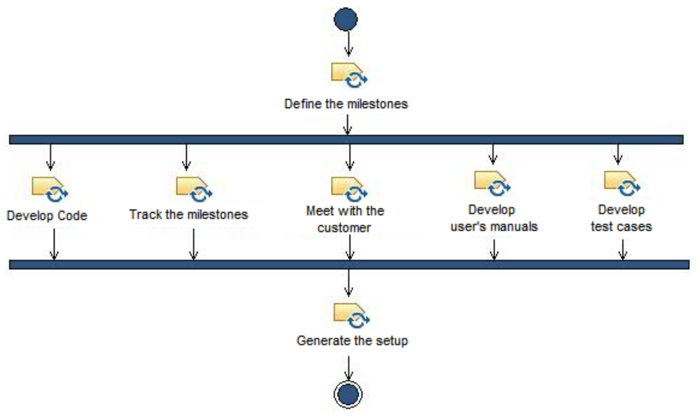

Fig. 7. *Implementation* process area after the *Complete Validation*

imented worker) cannot handle the complete picture even if it is about his/her own workflow. Finally, this evaluation stage gave the process engineers useful information to adjust once again the process in the EPFC. A new session was scheduled to validate the adjusted process to have the final approved version. Figure 7 depicts the final version of the *Implementation* process area after the *Complete Validation*. Three modifications were done from the previous workflow: two tasks were deleted (*Estimate equipment, time and tools* and the *Generate environment for development*) and a new one was added: *Develop test cases*. As in the Partial Validation session, the comments and observations, made by the process representatives, caused process engineers to make the necessary adjustments to the workflow. The comments and observations arose while the process representatives tried to follow the workflow simulating the execution of some real software projects they had to deal with.

4.3 Applying the Delivery phase

Finally, once having validated and approved the EPFC version of the software process, it was the time to leave it officially installed in the company (*Process Deployment*). One computer of the company was selected to leave the final version of the process installed. Next, the process engineers proceeded to first install the EPFC tool, then to copy the software process plug-in, and finally to generate the HTML view of the software process from the EPFC tool. To conclude it was made a brief last training session in the EPFC tool to the representative of the company who was designed to be in charge of the process for future evolution.

5 Conclusions

At least in the case of the application reported in the paper, the meta process resulted in a high quality process specification [14] that could be built in a short time. This issue reinforced our goal of proposing a light weight meta process even though more experimentation is still required.

In this paper we presented a software process formalization meta process and we described its application in a small Chilean company. Our experience has been much more satisfactory than in previous formalization endevours in similar settings. This is probably because the proposed meta process actually resulted light weight as planned for both process engineers and for people at the company as well. Agile principles[4] such as communication, simplicity, feedback and courage are all central to the meta process.

The whole formalization meta process lasted two months having two meetings per week among process engineers and people responsible for software development in the company. This time is quite short if compared to most reported case studies that declare to have invested no less than a year. The resulting formalized process still has certain underspecification [14], but they are neither too many nor difficult to overcome. In the opinion of the company people, this process will result in a high return of investment: the invested time was short, and as a payback they now count on an unambiguously defined, complete and agreed upon process. We are convinced that the success of the application was mostly due to the compromise of the CEO and the quality of the process engineers that took part on the meta process. More research is required for determining the applicability of the procedure with more precision.

The training activity motivated people in the company not only to understand what was being formalized, but also to value the possibility of easily maintaining and evolving the process [12]. This required a thorough knowledge about software processes and also about using the tool. As a corollary of the meta process application, we have realized that, even though people at the company participated actively during the whole formalization process, they required another training session about two months later in order to be able to completely adopt the formalized process and feel completely able to evolve it.

Acknowledgments

The work of Pablo Ruiz has been partly funded by Laccir. The work of Alcides Quispe, María Cecilia Bastarrica and Julio Ariel Hurtado Alegría has been partly funded by project Fondef D09I1171 of Conicyt, Chile.

References

1. Ulrike Becker-Kornstaedt. Towards Systematic Knowledge Elicitation for Descriptive Software Process Modeling. In Frank Bomarius and Seija Komi-Sirv, editors, *Third International Conference on Product Focused Software Process Improvement (PROFES '01)*, pages 312–325. Springer-Verlag, London, UK, 2001.

[4] Agile manifesto: http://agilemanifesto.org/

2. Ulrike Becker-Kornstaedt and Wolfgang Belau. Descriptive Process Modeling in an Industrial Environment Experience and Guidelines. In *Software Process Technology, 7th European Workshop, EWSPT 2000, Kaprun, Austria, February 21-25, 2000, Proceedings*, volume 1780 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2000.
3. Alberto Colombo, Ernesto Damiani, and Gabriele Gianini. Discovering the software process by means of stochastic workflow analysis. *Journal of Systems Architecture*, 11(52), November 2006.
4. Reidar Conradi, Christer Fernstrom, and Alfonso Fuggetta. A conceptual framework for evolving software processes. *SIGSOFT Software ACM Engineering Notes*, 18(4):24–35, October 1993.
5. Jonathan E. Cook and Alexander L. Wolf. Discovering models of software processes from event-based data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
6. Carlton A. Crabtree, Anthony F. Norcio, and Carolyn B. Seaman. An Empirical Characterization of the Accuracy of Software Process Elicitation. In Raffo et al. [18], pages 91–100.
7. Peter Feiler and Watts Humphrey. Software process development and enactment: Concepts and definitions. Technical Report CMU/SEI-92-TR-004, Software Engineering Institute, September 1992.
8. Javier García Guzmán, Yaser Rimawi, Maria Isabel Sánchez Segura, and Antonio de Amescua Seco. Ramala: A Knowledge Base for Software Process Improvement. In Ita Richardson, Pekka Abrahamsson, and Richard Messnarz, editors, *Software Process Improvement, 12th European Conference, EuroSPI 2005*, volume 3792 of *Lecture Notes in Computer Science*, pages 106–117. Springer, 2005.
9. J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, , and D. Zubrow. Benefits of CMM-Based Software Process Improvement: Initial Results. Technical Report CMU/SEI-94-TR-013, SEI, 1994.
10. Abram Hindle. Software Process Recovery: Recovering Process from Artifacts. In Giuliano Antoniol, Martin Pinzger, and Elliot J. Chikofsky, editors, *17th Working Conference on Reverse Engineering, WCRE 2010*, pages 305–308. IEEE Computer Society, 2010.
11. Dirk Höltje, Nazim H. Madhavji, Tilmann F. W. Bruckhaus, and Won-Kook Hong. Eliciting formal models of software engineering processes. In John E. Botsford, Ann Gawman, W. Morven Gentleman, Evelyn Kidd, Kelly A. Lyons, Jacob Slonim, and J. Howard Johnson, editors, *Conference of the Centre for Advanced Studies on Collaborative Research,(CASCON'1994)*, page 28. IBM, 1994.
12. Watts S. Humphrey. Software process improvement - a personal view: How it started and where it is going. *Software Process: Improvement and Practice*, 12(3):223–227, 2007.
13. Julio Ariel Hurtado, M. Cecilia Bastarrica, Alcides Quispe, and Sergio F. Ochoa. An MDE Approach to Software Process Tailoring. In Raffo et al. [18], pages 43–52.
14. Julio Ariel Hurtado, María Cecilia Bastarrica, and Alexandre Bergel. Analyzing software process models with AVISPA. In Raffo et al. [18], pages 23–32.
15. Maria Letizia Jaccheri, Gian Pietro Picco, and Patricia Lago. Eliciting software Process Models with the E^3 language. *ACM Transactions on Software Engeneering and Methodology*, 7(4):368–410, 1998.
16. Alejandro Lagos. Mejora Sistemática del Proceso de Desarrollo de Software de la Division de Sistemas de Autoservicio de la Empresa DTS (Spanish). Master's thesis, Computer Science Department, Universidad de Chile, 2011.

17. OMG. Software Process Engineering Metamodel SPEM 2.0 OMG Beta Specification. Technical Report ptc/07-11-01, OMG, 2007.
18. David Raffo, Dietmar Pfahl, and Li Zhang, editors. *International Conference on Software and Systems Process, ICSSP 2011, Honolulu, HI, USA, May 21-22, 2011, Proceedings.* ACM, 2011.
19. CMMI Product Team. CMMI for Development, Version 1.2. Technical Report CMU/SEI-2006-TR-008, Software Engineering Institute, 2006.
20. "JTC 1 Information technology /SC 7". ISO/IEC 12207:2008 Systems and Software Engineering – Software life cycle processes. Technical report, International Organization for Standarization ISO, 2008.
21. José Turgeon and Nazim H. Madhavji. A Systematic, View-Based Approach to Eliciting Process Models. In Carlo Montangero, editor, *Proceedings of the 5th European Workshop on Software Process Technology (EWSPT '96)*, pages 276–282. Springer-Verlag, London, UK, 1996.