A Practical Experience of a Software Process Line Creation

Andréa M Magdaleno[1], Renata M Araujo[2], Cláudia M L Werner[1], Carlos Freud Alves Batista[3]

[1]UFRJ – Federal University of Rio de Janeiro
COPPE – Systems Engineering and Computer Science Department
Zip 21945-970 – Rio de Janeiro – RJ – Brazil – P.O. Box 68511

[2]Graduate Program in Information Systems (PPGI) – UNIRIO

[3]Petrobras - TIC/CPSW/PGOD/PMSW

andrea@cos.ufrj.br, renata.araujo@uniriotec.br, werner@cos.ufrj.br, carlos.freud@petrobras.com.br

Abstract. Software process definition is a complex, time consuming and error prone activity. Such activity can be facilitated by a process reuse strategy. This strategy can be implemented through process line and components structures. This work presents a case study of a real process line creation in the context of an oil and gas company in Brazil. The results indicate both practical use and its potential to assess completeness and to identify inconsistencies in the organizational process.

Keywords: Software process, process definition, process composition, process line, process reuse.

1 Introduction

The assumption that a defined software process directly influences the quality of the developed product [1] has motivated many software organizations to invest in software processes definition. Process definition initiatives are usually challenged by **diversity** in various levels.

The diversity of **organizations** and **projects** makes the contexts in which processes are used very distinct. Since the universe of software projects is extensive and diversified, a single **software development model** cannot satisfy all of them. Despite plan-driven [2], agile [3] and free/open source software (FOSS) [4] development models aim to improve software development, each model represents a distinct development universe, differing in philosophy and main characteristics. In order to establish more effective software processes, research in the area has discussed how to promote the reconciliation among these development models [5].

Process diversity occurs when a project is executed by applying different

processes. It may happen: (i) concurrently in projects with different teams working in parallel; (ii) the adoption of different software processes throughout the lifecycle of a project, observed over time in the transition from development to maintenance stages [6]. Finally, a process cannot be defined without considering **people** who will use it (such as employees) or interact with it (such as customers and suppliers).

All these different kinds of organizations, projects, development models, people and teams make it harder to define specialized processes to cope with known and new development contexts [5]. There is a growing need for the effective definition of software processes that can handle this diversity [7].

Despite being one of the main tasks to be executed by the Software Engineering Processes Group (SEPG), process definition is not simple. Such activity demands experience and knowledge from several aspects of Software Engineering. Process definition can be time consuming, error prone and cause the following negative consequences [8]: unnecessary activities that lead to a waste of time and money; the omission of necessary activities, which may affect the product quality; and the failure to comply with the organizational or international standards.

In practice, process definition can be improved by a reuse strategy based on smaller process units. This work is based on process lines [7,9–11] and components. These reuse structures have the potential to deal with process diversity or variability, help to disseminate knowledge and successful experiences.

In order to facilitate process composition, we previously presented Context-Based Process Line Engineering (CBPL) [12] and COMPOOTIM [13] approaches to support project managers' decisions about process components selection and combination and optimize the suggestion of processes to the context of a particular project. Following our research agenda, the first analysis of a case study based on CBPL was discussed with the community in [14].

In this work, the case study's planning and execution are completely described. It was conducted in a real context of a large oil company in Rio de Janeiro, Brazil. The main goal was to evaluate the feasibility of supporting a process manager in the creation of the organizational process line. Process reuse is still a new topic in the company and this knowledge is not widespread. This case study is a step in this direction and a first version of the process line was achieved. Moreover, the company's interest in the topic shows that it is a real problem in the software industry.

The remainder of this paper is organized as follows. In Section 2, the CBPL approach is presented. Section 3 describes the plan and the execution of the case study. In Section 4, the results and threats to validity of the evaluation are analyzed. In Section 5, some related works are discussed. Finally, Section 6 concludes the paper and indicates some opportunities for future work.

2 Context-Based Process Line (CBPL)

A **process line** refers to the application of the product line [15] idea to processes. It corresponds to a set of processes in a particular domain, having common characteristics and being built based upon common, reusable process assets [11].

Aiming to offer a systematic approach to support process composition and reuse, a CBPL was created. It combines the reuse of the process line with the dynamic of

context [16], which makes possible to identify modifications in a process at runtime in order to adapt to new situations. CBPL corresponds to a set of process components, organized to represent common and variants parts within a specific domain that can be reused and combined, according to composition rules, to dynamically compose and adapt processes. The CBPL approach was similarly structured to the product line phases: Software Process Domain Engineering (SPDE) and Software Process Application Engineering (SPAE) (Fig. 1.) [10,12].

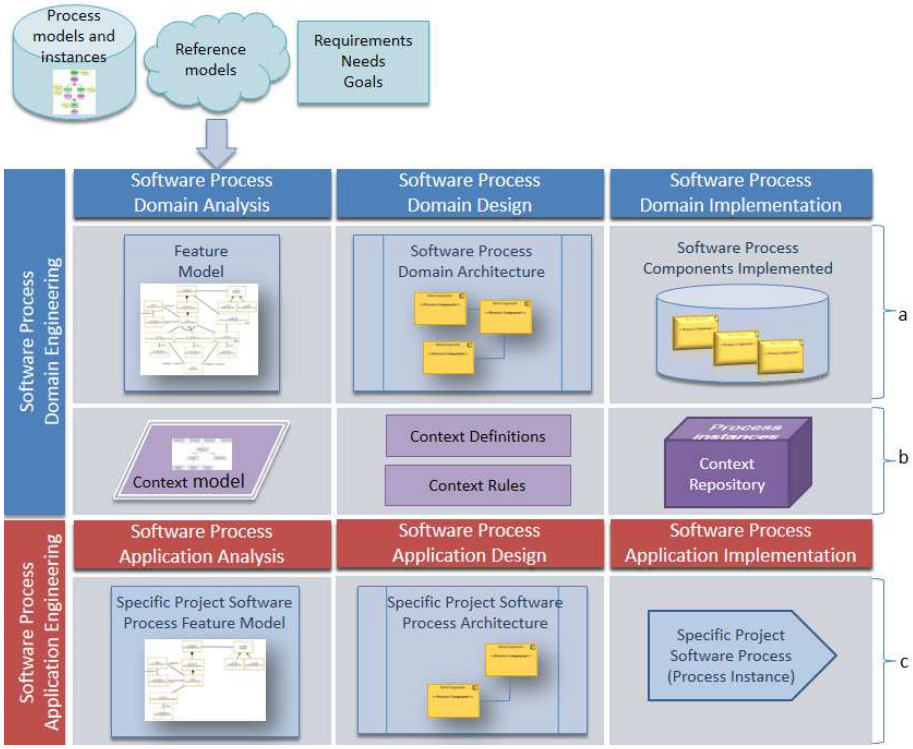

Fig. 1. Context-based process line (CBPL) approach [17]

SPDE is performed by *process engineer(s)* when there is the need to establish a process line; and when requirements, needs and goals change, promoting the evolution of the line. Existing process and/or reference models serve as input.

The SPDE phase creates the process line with a generic set of processes that captures the commonalities and variabilities in a domain. It makes explicit the points where these processes are similar and can be reused, and the points where they diverge, and need specific treatment. Variability in software processes becomes important in order to deal with diversity and adapt for use in a particular context.

In SPDE, the Analysis phase aims at identifying the domain knowledge through a *feature model* [18] (represents domain characteristics, commonalities, and variabilities of a process) and *composition rules* (represent dependencies or mutually exclusive relationships). At Design, *process architecture* is specified to define the main components, their internal properties and interfaces, and how they relate to each other. Each feature has a set of components that implement a process. *Implementation*

of the process architecture aims at generating executable models (Fig. 1.a).

Regarding context, during Analysis it is also possible to identify the contextual information considered relevant to describe the process. The *context model* is composed of context dimensions and information [19]. At Design, the context model is refined through *context definitions* (situations that may happen based on the combination of certain context information) and *context rules* (suggest process selection based on context definitions) [19]. Implementation involves the creation of a *context repository* (Fig. 1.b).

After establishing the SPL in SPDE, for each project, instead of defining a process from scratch, *project managers* can make use of the process line infrastructure to compose a specific project process in SPAE. SPAE phase contains activities similar to those presented in SPDE (Fig. 1.c). First, at Analysis, the outcome is a *cutting of the feature model*, containing only the features necessary for the new process. During Design, the process components are selected to be used within the architecture. From these inputs and the existing context, the new *process* is *composed*. Finally, Implementation comprises an analysis of supporting tools for process execution. As a result, the *process* is ready to run and to be adapted at run time.

This work focuses on the organizational SPL creation at SPDE. Within SPDE, the key phases of this research are Analysis and Design, because it is precisely during these two phases that process managers need assistance to deal with variability.

3 Case Study

In this section, we discuss the creation of a SPL in a real industrial context in Brazil.

3.1 Planning the study

This case study has the following goal and scope, defined according to [20] structure:

Analyze the CBPL approach
With the purpose of characterizing
With respect to feasibility of creating a SPL
From the point of view of a manager of the SEPG
In the context of a real industry environment

This case study used data from organizational process models and standards. These documents describe in details the activities, roles and artifacts of the software process. One development methodology - Object Orientation - was specifically selected among the approximately 12 existing methodologies in the organization. Traditional project management, the project management with agile methods and Testing and Configuration Management subprocesses were also adopted.

Five main instruments were designed and validated with an expert: the *term of consent* that declares the purpose of the study and ensures data confidentiality; the *characterization form* to determine the participant's profile; *training material* used to explain the main concepts of SPL; the *study form* used for collecting the results of the tasks; and a *survey*, which intended to obtain qualitative information about the study.

3.2 Conducting the study

This case study was performed in March 2013 with the manager of SEPG (composed by 20 members and located in a specialized division of the organization focused on software processes and methodologies). The profile of the participant indicates 10 years of experience in Software Engineering, Project Management and Software Process Definition. However, the participant did not have experience in Reusing Software Processes. The participant claimed to have much familiarity with all of organization's software processes. These processes were characterized by him as large, with an average complexity and with high relevance to the organization.

From the company's original processes (which cannot be made available due to a non-disclosure agreement), an initial draft of the SPL was created following the steps of SPDE. The resulting SPL was presented to the SEPG manager and evaluated based on his experience and knowledge. Each session analyzed different artifacts and suggestions and comments of the expert were recorded on a spreadsheet.

In the first step, the main reference for the creation of the features model was the organizational software development process. From this process, 86 process features were extracted – including activities, tasks and products (Table 1). The features were organized by phases (initiation, planning, execution, monitoring and closure).

Table 1. SPL' features

Phase	Total	By type		By Optionality		By Variability	
Inception	**26**	Activity	1	Mandatory	8	Invariant	26
		Task	9	Optional	18	Variant	0
		Product	16			Var Point	0
Planning	**14**	Activity	3	Mandatory	1	Invariant	11
		Task	5	Optional	13	Variant	2
		Product	6			Var Point	1
Execution	**33**	Activity	4	Mandatory	8	Invariant	30
		Task	15	Optional	25	Variant	2
		Product	14			Var Point	1
Monitor and Control	**9**	Activity	1	Mandatory	3	Invariant	6
		Task	2	Optional	6	Variant	2
		Product	6			Var Point	1
Finishing	**4**	Activity	1	Mandatory	4	Invariant	4
		Task	1	Optional	0	Variant	0
		Product	2			Var Point	0

The *process features model* of the planning phase is partially presented in Fig. 2. This model uses *OdysseyProcess-FEX* notation [21] and was created with the support of the Odyssey environment [22]. It is composed by 3 activities, 4 tasks, and 10 products. These elements are related through different types of relationships. The mandatory features are represented by a continuous line and optional features by a dotted line. This feature model has two main activities: Traditional and Agile Planning, both optional and composed by tasks with different planning granularity. In the Traditional approach, it indicates both project and solution planning. In the Agile approach, it is suggested both the release and the sprint planning.

From the process features, 8 composition rules were created. The features

mentioned in the rules are indicated with a RCCAX code in the upper right corner (Fig. 2). Some of these rules are presented in Table 2.

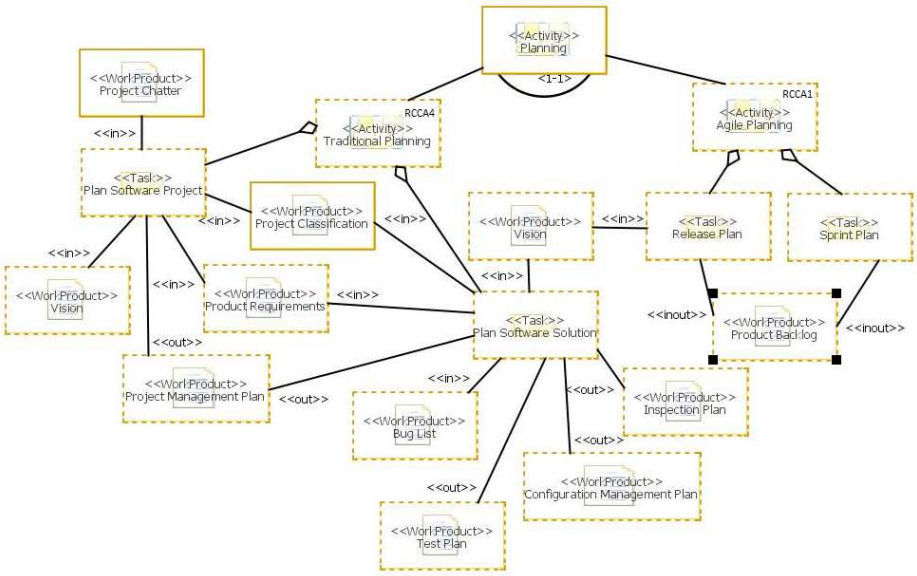

Fig. 2. Process features – Planning phase.
Legend: Mandatory process features – Rectangle with continuous line. Optional process features – Rectangle with dotted line. Composition rules – Code in the upper right corner

In SPDE, it is also required to define the *context model*, including context dimensions and information. As a starting point, Araujo *et al*. [23] suggest nine context dimensions for the software domain. In this study, these context dimensions were maintained, because the company adopts no specific definition.

Table 2. Examples of features composition rules

# Rule	Antecedent	Type	Consequent
RCCA1	Agile planning OR Build agile software solution	*Inclusive*	Track agile project
RCCA8	Request approval of project vision	*Inclusive*	Develop project vision

The 24 context information items identified were distributed on 4 of these dimensions: customer/user, software product, project, and team (Table 3). The context information elements have been defined based on the criteria for rating the complexity of the project and assumptions for agile development (Table 3). To amplify the understanding, we can take one of them as example: Type of Development. The main idea is that the company can have process variations for corrective, emergency, evolutionary/adaptive maintenance, or new development.

Table 3. Context dimensions and information

Dimension	Context Information
Customer/User	Geographical dispersion.
Software	Solution with a deadline of use, Impact of unavailability, Use of

Dimension	Context Information
Product	standard infrastructure, Availability of documentation, Algorithmic complexity, Components reuse, Complexity of the logical data model, Amount of concurrent users, and Information classification.
Project	Scope instability, Deadline constraints, Need for procurement or contracting, Sprints (number and duration), Use of new technologies, Type of development, HH amount, and Accordance with SOX.
Team	Team size and Solution knowledge.

In the next step, 86 process components (Table 4) were extracted detailing the 32 features of type "Task" created in the previous phase of SPL. For instance, from the feature “Plan Software Project” (Fig. 2), 11 process components following the traditional project management methodology were detailed. From the process components, 9 composition rules were identified.

Table 4. SPL‘ process components

Methodology/Subprocess	Total	Mandatory	Optional
Traditional project management	29	12	17
Agile project management	8	4	4
Object Orientation	21	1	20
Software Test	5	4	1
Software Configuration Management	12	8	4
General	11	3	8
Total	**86**	**32**	**54**

Table 5 shows an example of a process component. The structure of a process component in this research included: i) identifier; ii) name; iii) role; iv) description; v) associated feature; vi) optionality; vii) variability; and viii) input and output artifacts.

Table 5. Process component example

Process Component: CP01. Project Responsible Assignment	
Role: Team Leader	
Description: Project start milestone. Formally designate the leader of the project to form the Project Open Charter.	
Process Features: Start project	
Optionality: Mandatory	**Variability:** Invariant
Input:	Software Project Information *(Mandatory)*
Output:	Project Open Charter

Regarding the context model, after understanding context information, the next step was to establish context definitions. Context definitions represent circumstances that may happen based on the combination of values of certain context information. A total of 26 context definitions were identified. An example of a common situation that influences the organizational software process domain is “Agile project” defined by the expression: Sprints=Yes AND (Number of Sprints >=2) AND (Team Size<=9 members) AND (Sprints Duration=1<=x<=4). An expression combines the values of the previously defined context information (Table 3).

Context rules can specify the actions to be taken for a given situation. They represent how a context situation affects the configuration of a software process. The

context rules were generated based on context situations and features model of the SPL (Table 6). It is noteworthy that no context rules were defined by setting actions for all context definitions identified, which means that in practice some definitions will not be applied in this case study. An increased use of context situations and rules in SPL would increase the wealth of suggestions that CBPL is able to make in the composition of software processes.

Table 6. Examples of context rules

# rule	Context Definition		Features
RCTX1	Agile project	*Implies*	Agile Planning AND Track agile project
RCTX3	New development OR Nonstandard infrastructure OR Project with acquisitions	*Implies*	Assess feasibility of infrastructure
RCTX4	SOX Project	*Implies*	Inspection
RCTX5	Complex modeling	*Implies*	Data administration

At the end of this study, a complete SPL was ready for use in the organization. The discussion of the main results is presented in the next section.

4 Results Analysis

The study was performed in four individual sessions, spread over a month and with a mean duration of 01:20 hs each. During the sessions, the created SPL was presented to the SEPG manager and evaluated by him using the peer review technique. This review resulted in 44 comments.

These comments were classified according to the severity, phase of the development process, type and details of artifact where the remark was made, and classification based on defect taxonomy [24]. The analysis of the defects found, summarized by the graphs in Fig. 3, allows us to observe that 60% of the defects has low severity. Only three defects had high severity, as follows:

- In the case of agile method, creating activities features (representing the stages of development) was not explicitly presented in the original process model. However, without using such an aggregator, the complexity of the features model increases and its legibility would be compromised. Therefore, a decision had to be taken about complexity versus reliability in the representation. In the final model, the subject chose to keep the division in stages;
- When looking at the process features diagram, the participant missed a representation of the process activities sequence. However, the notation adopted [21] does not provide this kind of representation;
- Finally, the participant noted that the same artifact is an input for many tasks. Therefore, it could be put as input in the activity that comprises all these tasks, rather than being repeated for all of them.

According to the classification applied (Fig. 3), only 1 defect (regarding agile method) was rated as "incorrect fact", as explained above. The 11 "omission" defects were caused by the absence of any item in the SPL or by the lack of information about the mandatory elements in the process documentation. Most of the defects were

caused by "ambiguities" or "inconsistencies" between the organizational processes and procedures that required interpretation by the SEPG manager during the execution of the study. The comments classified as "new information" relate to items that did not appear in the documentation originally submitted and were added as supplementary materials by the SEPG manager during the study.

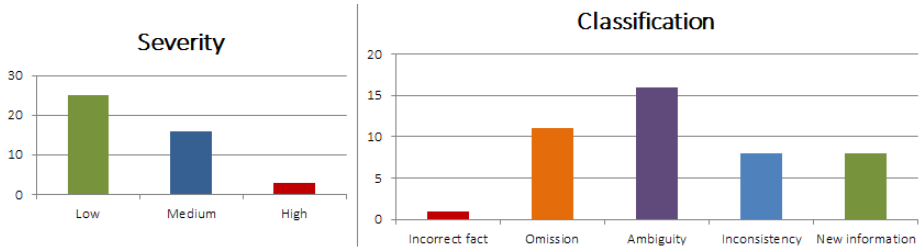

Fig. 3. Analysis of defects

4.1 Qualitative Observations

The subject affirmed that he was able to perform all tasks and was satisfied with the final SPL. It was considered that the resulting SPL had a large size, due to the amount of artifacts that comprise it, but noted: "*I have never seen another process line to compare the size*". He further stated that the resulting line had a high completeness and that the time for creation/evaluation of the SPL seemed reasonable.

Although the initial SPL has been created by the researcher, during the review the subject followed all the steps and reviewed all artifacts of the SPL. The participant rated this revision as medium degree of difficulty and stated "*it required some interpretations*". These interpretations concern the exercise of fitting the process that currently exists at the organization in the new form of representation proposed considering reuse. The current process modeling notation was EPC (Event-driven Process Chain) [25] and as mentioned the creation of the SPL adopted OdysseyProcess-FEX notation [21] based on features modeling.

When asked about the main **difficulty** during the evaluation of the SPL, the participant highlighted the difficulties of understanding the need for composition rules, as some of them seem obvious and could be replaced by process components sequencing. However, it is also worth noting that the organization's processes do not provide a high level of variability. This is because, currently, the definition of business processes is based on the adoption of a generic standard process. It is not desired by the company to overload project managers with many decisions during the tailoring of a process specific to the project. Therefore, the process has a low level of flexibility, which limits the acting of composition rules. COMPOOTIM approach brings another perspective, since it can empower project managers' decisions.

It was also examined whether the CBPL contributed to make **optionality** and **variability** explicit. The participant indicated that they were explicit, but also demonstrated worry about the full representation of the company's software process: "*If the complete process of the organization was mapped, it would not guarantee the correctness of the model without the support of a tool*". This comment can be understood considering that only one methodology (Object Oriented) was selected,

among the 12 currently adopted by the organization. With the modeling of other methodologies, the generated SPL would be bigger and demand a support tool.

Considering the experience of the participant as the SEPG manager, one can imagine that he can judge the feasibility of practical use of the SPL created in the organization. The participant stated that the SPL could be adopted in the organization with adjustments. He suggested to include the possibility to choose the artifacts considering the particular needs of each organization. This possibility would need to be investigated in future evaluations. Still referring to the practical use of the SPL, he spontaneously highlighted: "*I already requested the features model for use in my daily work in the company*". This review highlights a practical contribution for organizational discussions of the processes.

The main advantage identified by the participant was the fact that the systematic approach helps to assess the completeness and identify inconsistencies in the process. This benefit can be considered an unexpected side effect, because it had not been foreseen during the CBPL proposition. In addition, the approach was considered helpful for novices or low experience project managers: "It *helps even those who do not have much knowledge about software engineering*".

On the other hand, the main challenge mentioned was the difficulty in maintaining the SPL updated, as business processes change frequently. Since there are many traceability among artifacts, the maintenance could be costly if changes have to be propagated manually between different artifacts. However, the computational support available through COMPOOTIM tool [12] can facilitate this maintenance.

In addition, one negative aspect mentioned was the lack of simplicity of the systematic approach that requires assistance (from the researcher and tool). The participant explained: "It is *laborious because it is a new approach that adopts a different notation*". The need for a researcher support comes from "*difficulty in acquiring new knowledge to do it yourself when you're already used to work differently*". This comment refers to the difficulty of paradigm change, since organizations typically adopt process notations (such as BPMN - Business Process Management Notation [26], EPC [25]). Moreover, the structures of process reuse are new to the participant. The demand for computational support is natural and expected.

4.2 Threats to Validity

This section discusses the main threats to validity for this study, following Wohlin *et al*. [20] classification.

The main *conclusion threat* of this study is the sample size, with only one SPL using processes from a single company considering one participant point of view, not being ideal from a statistical standpoint. Therefore, this study presents a limitation in the results that will be considered only as initial evidence. However, we must consider that this is one important oil company in Brazil with national geographic distribution. In addition, actual organization software processes based on different software methodologies were used to create the SPL with some variability to represent plan-driven and agile software development models diversity.

As the subject was chosen for convenience, it is a *construct threat*. It is possible that his behavior was based on assumptions about the expected results from this study. A random selection of participants was not possible since participants with

knowledge of the company's software process and with experience in software processes definition were required. Another threat to validity is that the SPL has been created by the researcher and have been validated by the participant. Despite having followed all the steps of the CBPL during this evaluation, the participant did not actually create the SPL from the beginning.

Regarding *internal threats*, we should consider that it was a long study distributed in many sessions. To prevent the subject from being tired or discouraged, the study was conducted in more than one session.

Finally, a *threat to external validity* is that the experiment only considers the SPL from a single company and from the point of view of a single subject. Thus, it is not possible to generalize the results to other contexts, but it was a first step in a practical SPL creation in the Brazilian industry context.

5 Related Work

Although many of the proposals in the SPL area [10,11,27–31] do not present experimental studies or validations, some exceptions [7,9] should be mentioned. Aleixo *et al*. [7] evaluate the feasibility of their proposal, implementing it by using several model-driven technologies. However, there was no application to an industrial scenario. In [9], the authors use a previous SPL creation experience to detail their approach. The created SPL focused on the acquisition processes and was created and evaluated by internal specialists. Therefore, both results cannot be compared to the results obtained on the present work. Other different approaches are analyzed below.

Process Family Engineering [30] provides techniques for enabling the production of processes in a certain business, where each product represents a set of processes enabled at a certain moment of the execution. This approach produces one software system that evolves at runtime, where the features are processes.

Rombach [10] states that processes within an organization could be organized according to similarities and differences and presents SPPL (software product and process line) engineering where appropriate artifacts and processes can be chosen based on a set of product and process requirements according to project constraints.

Washizaki [11] proposes a process-tailoring technique which intends to solve problems with component-based and generator approaches by building a Process-Line Architecture (PLA). The goal is to derive project-specific processes from the PLA, combining, extending and reusing core processes and their variants for a particular problem domain, following a "bottom-up" technique that uses existing knowledge on process definitions and applications in the problem domain.

Finally, other two closely related areas are contingency factors and Situational Method Engineering (SME). Contingency factors [32] refer to variables that characterize a project to determine the selection of an appropriate method from a portfolio. This idea of project characterization is similar to the one proposed in this work. The area of SME [33] is intended to provide organizations with flexibility in configuring a project-specific process, using methods or fragments stored in a repository. This support and the existence of a range of process knowledge are similar to the CBPL approach. However, project manager needs assistance to make decisions about the process to be adopted in the project. SME gives no indication about the

selection and combination of them to guarantee that the resulting process is completely usable. The capacity to compose this process, considering the project context, is a contribution of COMPOOTIM approach.

6 Conclusion

This paper presented the CBPL approach to software processes composition through process lines. This approach was evaluated through an experimental study conducted in the context of a large oil and gas company in Brazil. The motivation of the company in providing a professional at management level to participate in this study shows that it is an actual problem in the software industry.

The objective was to evaluate the feasibility of creating a process line. The main result was a complete SPL that makes sense to the manager of SEPG. This SPL can be used in the future both by the organization to the definition of internal software processes as well as researches in other experimental studies.

The main conclusions obtained during this case study concern the practical use of the SPL, which contributed to the organizational discussions of the processes and to assess the completeness and identify inconsistencies in the process. The approach was considered helpful for novices or low experience project managers. The next steps for the generalized adoption of the SPL in the company should be: to train the members of SEPG in the concepts of SPL and variability; expand the SPL to include other methodologies; and plan the implementation of a support tool such as COMPOOTIM.

Although this study has obtained positive results, it is noteworthy that this is only an initial evidence, due to the limited number of participants and the use of only part of the processes of the organization. However, the company's interest in maintaining a joint research effort represents a real opportunity for future expansion of process reuse initiatives in the organization. In addition, future replications of this study both inside the company (internal) and with other companies (external) can be conducted to verify if a new effort to define an SPL will succeed.

Moreover, the case study indicated the need for a future review on the need of the composition rules, considering the alternative of mechanisms to sequence process components. After this review, one can verify if this systematic can be simplified or adapted according to the needs of use and representation of each organization while maintaining the desired results. Other mechanisms that facilitate process consistency check is currently under development as part of our research group work with SPL.

Finally, this study also indicated the need of some improvements on the OdysseyProcess-FEX notation. For example, difficulties with the multiplicities in the activities features were observed. Therefore, the use of other notations, such as [34], can be helpful and should be evaluated in future experimental studies. Another possibility is to transform the final artifacts to a BPMN notation that should be more user-friendly to the SEPG.

Acknowledgments

This work is partially funded by CNPq, FAPERJ and CAPES.

References

[1] A. Fuggetta, “Software process: a roadmap”, *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland: ACM, 2000, pp. 25–34.

[2] M.B. Chrissis, M. Konrad, and S. Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*, Boston, USA: Addison-Wesley, 2006.

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, et al., “Manifesto for Agile Software Development,” 2001.

[4] E.S. Raymond, *The Cathedral & the Bazaar*, O’Reilly Media, 2001.

[5] A.M. Magdaleno, C.M.L. Werner, and R.M. Araujo, “Reconciling Software Development Models: A Quasi-Systematic Review,” *Journal of Systems and Software (JSS)*, vol. 85, 2012, pp. 351–369.

[6] N.T. Siebel, S. Cook, M. Satpathy, and D. Rodríguez, “Latitudinal and longitudinal process diversity,” *Journal of Software Maintenance: Research and Practice*, vol. 15, Jan. 2003, pp. 9–25.

[7] F.A. Aleixo, M.A. Freire, W.C. Santos, and U. Kulesza, “A Model-Driven Approach to Managing and Customizing Software Process Variabilities,” *International Conference on Enterprise Information Systems (ICEIS)*, Funchal, Madeira, Portugal: 2010, pp. 92–100.

[8] O. Pedreira, M. Piattini, M.R. Luaces, and N.R. Brisaboa, “A systematic review of software process tailoring,” *SIGSOFT Software Engineering Notes*, vol. 32, 2007, pp. 1–6.

[9] A. Barreto, A.R. Rocha, and L. Murta, “Supporting the Definition of Software Processes at Consulting Organizations via Software Process Lines,” *International Conference on the Quality of Information and Communications Technology (QUATIC)*, Porto, Portugal: 2010, pp. 15–24.

[10] D. Rombach, “Integrated Software Process and Product Lines,” *Unifying the Software Process Spectrum*, Heidelberg: Springer-Verlag, 2006, pp. 83–90.

[11] H. Washizaki, “Building Software Process Line Architectures from Bottom Up,” *Product-Focused Software Process Improvement (PROFES)*, Amsterdam, The Netherlands: LNCS, 2006, pp. 415–421.

[12] A.M. Magdaleno, R.M. Araujo, and C.M.L. Werner, “COMPOOTIM: An Approach to Software Processes Composition and Optimization”, Congresso Ibero-Americano em Engenharia de Software (CIbSE), Buenos Aires, Argentina: 2012, pp. 1–14.

[13] A.M. Magdaleno, M. de O. Barros, C.M.L. Werner, R.M. de Araujo, and C.F.A. Batista, “Collaboration Optimization in Software Process Composition,” *Journal of Systems and Software (JSS) - Special Issue Search Based Software Engineering (SBSE)*, 2014 (to appear).

[14] C.F.A. Batista, A.M. Magdaleno, C.M.L. Werner, and R.M. de Araujo, “A Experiência da Petrobras na Criação de uma Linha de Processos de Software”, Brasília, DF, Brasil (In Portuguese): 2013.

[15] L.M. Northrop, “SEI’s software product line tenets,” *IEEE Software*, vol. 19, 2002, pp. 32–40.

[16] P. Brezillon, “Context in problem solving: a survey,” *Knowledge Engineering Review*, vol. 14, 1999, pp. 47–80.

[17] V.T. Nunes, C. Werner, and F.M. Santoro, “Context-Based Process Line,” *International Conference on Enterprise Information Systems (ICEIS)*, Funchal, Madeira, Portugal: 2010, pp. 277–282.

[18] K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson, *Feature-Oriented Domain Analysis*, CMU-SEI, 1990.

[19] P. Fernandes, C. Werner, and L.G.P. Murta, “Feature modeling for context-aware software product lines,” *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2008, pp. 758–763.

[20] C. Wohlin, P. Runeson, and M. Höst, *Experimentation in Software Engineering: An Introduction*, Springer, 1999.

[21] E.N. Teixeira, “OdysseyProcess-FEX: Uma Abordagem para Modelagem de Variabilidades de Linha de Processos de Software,” 2011.

[22] ODYSSEY, “Odyssey SDE Homepage,” 2014.

[23] R.M. de Araujo, F.M. Santoro, P. Brézillon, *et al.*, “Context Models for Managing Collaborative Software Development Knowledge,” *Workshop on Modeling and Retrieval of Context (MRC)*, Ulm: 2004, pp. 61–72.

[24] R.M. de Mello, E.N. Teixeira, M. Schots, *et al.*, “Checklist-Based Inspection Technique for Feature Models Review,” *2012 Sixth Brazilian Symposium on Software Components, Architectures and Reuse*, Los Alamitos, USA: IEEE Computer Society, 2012, pp. 140–149.

[25] A.-W. Scheer, *ARIS - Business Process Modeling*, Springer, 2000.

[26] OMG, “Business Process Management Notation (BPMN) Version 1.2,” 2009.

[27] O. Armbrust, M. Katahira, Y. Miyamoto, J. Münch, H. Nakao, and A. Ocampo, “Scoping Software Process Models - Initial Concepts and Experience from Defining Space Standards,” *Making Globally Distributed Software Development a Success Story*, Berlin / Heidelberg: Springer, 2008, pp. 160–172.

[28] I. Montero, J. Pena, and A. Ruiz-Cortés, “Business Family Engineering: Does it make sense?,” *I JISBD Taller sobre Procesos de Negocio e Ingeniería del Software (PNIS)*, Zaragoza, España: 2007, pp. 34–40.

[29] B. Simidchieva, L. Clarke, and L. Osterweil, “Representing Process Variation with a Process Family,” *Software Process Dynamics and Agility*, Minneapolis, MN, USA: Springer, 2007, pp. 109–120.

[30] J. Bayer, W. Buhl, C. Giese, T. Lehner, A. Ocampo, F. Puhlmann, et al., *Process family engineering. Modeling variant rich processes*, 2005.

[31] J.A.H. Alegria and M.C. Bastarrica, “Building software process lines with CASPER,” 2012, pp. 170–179.

[32] W. Bekkers, I. van de Weerd, S. Brinkkemper, and A. Mahieu, “The Influence of Situational Factors in Software Product Management: An Empirical Study,” *International Workshop on Software Product Management (IWSPM)*, Barcelona, Catalonia, Spain: 2008, pp. 41–48.

[33] S. Brinkkemper, “Method engineering: engineering of information systems development methods and tools,” *Information and Software Technology (IST)*, vol. 38, 1996, pp. 275–280.

[34] K. Czarnecki, S. Helsen, and U. Eisenecker, “Formalizing cardinality-based feature models and their specialization,” *Software Process: Improvement and Practice*, 2005, p. 2005.