Broadening the Use of Process Patterns for Modeling Processes

Hanh Nhi Tran, Bernard Coulette,
University of Toulouse 2, 5 allées A. Machado, F-31058 Toulouse, France
{tran,coulette}@univ-tlse2.fr

Bich Thuy Dong
University of Natural Sciences, 227 Nguyen Van Cu, Q5, HoChiMinh Ville, Vietnam
thuy@hcmuns.edu.vn

Abstract

Generally, process patterns are considered as patterns capturing reusable development activities, and serve as building blocks for constructing new processes. However, such a definition is not adequate to represent the original idea of process patterns that aims to capture and reuse diverse process knowledge. In this work, we broaden the definition and application of process patterns to take more advantage of them for different process modeling needs. First, we formalized the process pattern concept so that it can capture various kinds of process knowledge. Then, we proposed different ways for reusing process patterns to generate or improve process models. These propositions were rigorously defined in a UML-based meta-model to permit describing processes based on process patterns with standard notations, and to facilitate the development of supporting tools.

1. Introduction

Software Process Modeling deals with producing explicit, formal representations of software processes to help understand, analyze, enact and improve software development. Because software processes are intrinsically complex, process modeling is one of the most fundamental challenges of Software Process Technology. Reuse of the valuable process knowledge gained through the modeling of software processes is thus important to reduce process modeling effort and propagate process best-practices.

Inspired by the success of software product patterns, the process community proposed the concept of process pattern to capture and reuse explicitly process knowledge. Unfortunately, this attractive concept has still been poorly exploited due to the limited definition, the inadequate formalization and the lack of supporting methodology.

Generally, process patterns are defined as patterns describing proven processes for realizing development tasks (e.g. the *Technical-Review* pattern in [1] describes how to organize and conduct the review of one or more deliverables). Defined in this way, process patterns are only used as building blocks for constructing new processes. However, process knowledge of interest includes not only such software development tactics, but also other heuristics of process modeling, organizing and execution (e.g. the patterns proposed in [16] define pragmatic and abstract building blocks for modeling recurrent situations in collaborative works). This kind of process knowledge can be useful for both process construction and process improvement, but it is currently ignored in process pattern formalization and application.

We argue that the existing definitions of process patterns are not enough general to cover the diversity of process knowledge. Based on these inadequate definitions, the ways that process patterns are formalized and used are limited too. We think that broadening the view on process patterns can help us taking more advantage of process patterns for different process modeling needs.

Thus, we present in this paper a meta-model that enables the representation of diverse process patterns and their applications for elaborating process models based on process patterns.

The second section gives a brief survey of some significant works on process patterns and contrasts them with our approach. The third section presents how our process pattern meta-model supports capturing and representing various process knowledge. The fourth section describes how process patterns can be used to generate or improve process models. The conclusion sums up our contributions, and future works.

2. State-of-art

In this section we want to summarize what kinds of process knowledge are generally captured in published process patterns and how those process patterns are formalized and reused by the existing works for facilitate process modeling.

We classified process knowledge into three principal types according to the level of abstraction: (a) generic process structures applicable to any process, such as templates for modeling different process workflows [29]; (b) general development methods or techniques that can be applied for different application domains, such as the inspection technique for verify software artifacts [5]; (c) concrete hands-on experiences applicable to a particular process for producing a specific kind of product, such as a process for designing information systems adopted by a specific team.

Most of process-related patterns in literature capture the process knowledge type (b) [1,5,6,8,7] or (a) [16,21,22,29]. The knowledge type (c) is normally captured in internal process patterns of an organization.

There is few works on process patterns formalizing; we describe some of those that define a meta-model for process pattern.

In the Living Process Framework [9], Gnatz et al. introduced the notion of process pattern as a modular way for documenting development knowledge. However, the process pattern structure as well as interrelationships of process patterns is not defined explicitly in their work. In contrast, Hagen et al. developed a UML-based language named PROPELS [10] to describe explicitly process patterns. The meta-model of PROPELS encloses the internal structure as well as relationships between process patterns to facilitate patterns organization. Ribo et al. [7] developed the language PROMENADE which defines process pattern as a process template and provides a set of operators supporting process reuse and harvesting mechanisms to manipulate process models.

In the draft submission for SPEM 2.0 [18], a process pattern describes reusable clusters of activities in common process areas. Pattern applications is supported through the Activity Use mechanisms, which defines relationship kinds so that when the pattern is being revised or updated, all changes will automatically be reflected in all processes that applied that pattern. However, the internal structure of this concept and relationships between patterns are not defined in SPEM

All the above works adopted a rather limited definition for process pattern concept, i.e. they focused mainly on capturing process knowledge type (b) and (c). They proposed a unique way to reuse process patterns as building blocks for generating new development processes.

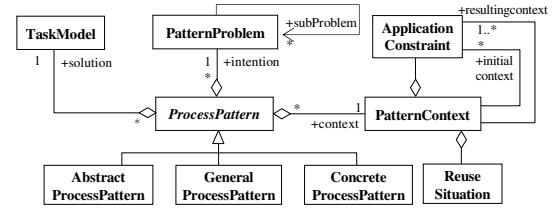
In contrast to these works, we define process patterns as patterns for modeling processes. Thus, the process patterns in our definition can capture diverse process knowledge in all (a), (b) and (c) (c.f. [27] for a more complete classification of process patterns). We want to provide different mechanisms to apply process patterns not only for elaborating new process models, but also for redesigning or improving existing ones.

3. Process Patterns Formalization

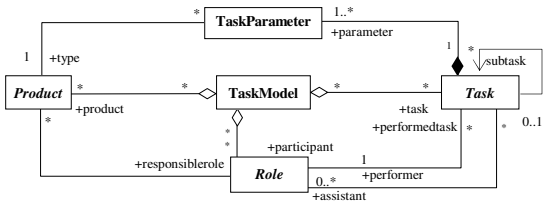
To enable a practical use of process pattern in process modeling, we formalize this concept by introducing it into a process meta-model. In order to meet the standardization and facilitate the development of supporting tools, our meta-model is inspired by SPEM1.1[17] and defined as a MOF-based[20] model by reusing the UML 2 Infrastructure library [19].

In our meta-model, a process pattern (*ProcessPattern*) captures a model (*TaskModel*) representing a solution that can be applied in a given context (*PatternContext*) to resolve a recurrent process modeling problem (*PatternProblem*). (Figure 1a).

The *TaskModel* concept is used to describe a (fragment of) process and is an aggregation of a set of process elements, i.e. *Task*, *Product* and *Role* (Figure 1b). A *Task* is a unit of controlled work (i.e. scheduled) realized by a *Role* to create or modify *Products*.



(1a) Meta-model for process patterns



(1b) A TaskModel represents the solution of a process pattern

Figure 1 Meta-model of process pattern concept

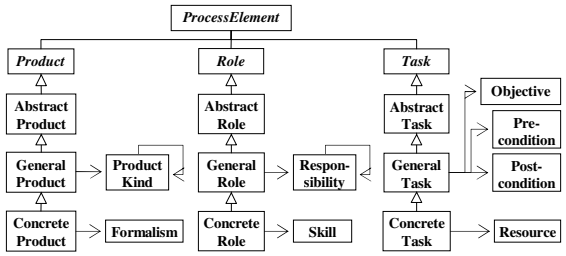
Process modeling problems can vary from describing specific development tasks to suggesting efficient generic process structures. Consequently, solutions provided by process patterns could describe diverse process knowledge at different levels of abstraction.

We distinguish three kinds of process patterns: *abstract*, *general* and *concrete* to capture respectively three process knowledge type (a)(b)(c) mentioned in section 1. An *AbstractProcessPattern* captures a generic recurrent structure for modeling or organizing processes. In such a pattern, the precise semantic of elements is not important but the relation between them

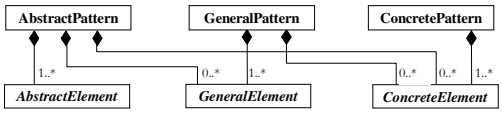
express the solution. For instance, in [16] Lonchamp described the pattern *"Division of labour"* as a solution for modeling collaborative tasks performed by several actors on subparts of a document for building the whole document in a parallel way (Figure 3a) . A *GeneralProcessPattern* provides a general development method which partially specified and must be refined further to be applicable for a specific purpose. For example, Figure 3b shows the *Fagan inspection process* [5] for detecting defects of software artifacts. The task of this process must be specialized when applying for a specific kind of products (e.g. requirements, design, code, etc.); therefore we use a general pattern to describe it. Finally, a *ConcreteProcessPattern* captures a completely specified solution of a process in a particular context. For example, Figure 3c describes a particular process adopted at the *Vietnamese University of Natural Sciences* (NSU) for design information systems for development teams having one developer and one tester. The tasks of this process are specified into detailed steps, products according to the development approach and technique used at NSU (e.g. RUP-based process, relational database design) thus concrete.

To support representing process patterns at the above levels of abstraction, we also describe process elements at three abstraction levels: *abstract, general* and *concrete*.

We show in Figure 2 the hierarchy of process elements (a), and the possible compositions of process patterns on different levels of abstraction (b).



(2a) Definition of process elements on different levels of abstraction

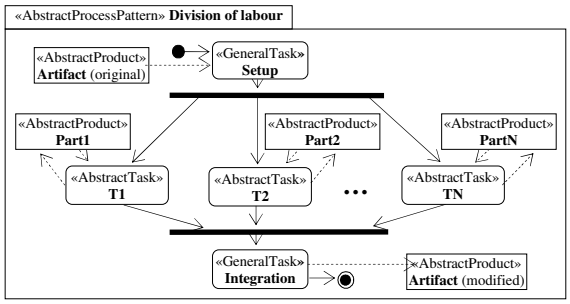


(2b) Compositions of process patterns on different abstraction levels

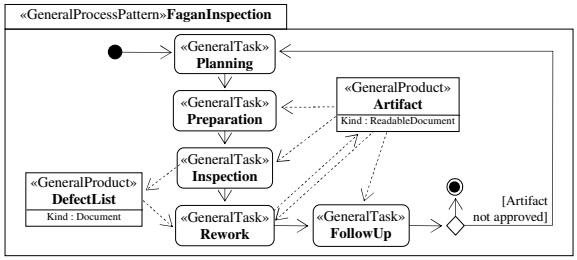
Figure 2 Abstraction levels of Process Elements

An *abstract process element* is used to refer to an unspecified element without defining its precise semantic (e.g. a task *T*, a product *P*). A *general process element* is partly defined but could be specified further (e.g. a *code product* can be object-oriented or functional with different characteristics; a *review task* can work on code or design document with different details, techniques). A *concrete process element* is completely specified for a specific purpose (e.g. a *UMLDesignReview* task).

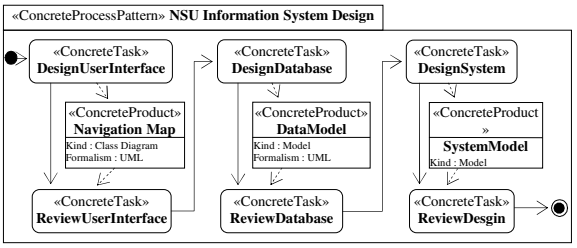
Figure 3 shows an example of different process patterns in our categorization.



(3a) A template structure enabling parallel collaborative works (adapted from [16])



(3b) Fagan method for inspect defects of an software artifact (adapted from [5])



(3c) The Design Process of NSU team having 1 designer and 1 tester

Figure 3 Example of three kinds of process patterns

3. Applications of Process Patterns

After examining many issues in process modeling, we distinguish two types of problems: *defining a process element* and *organizing process elements*.

The first problem concerns the question of *how to define the details of a process element* (including development tasks, work products or participant roles). A pattern addressing such a question captures a model describing the content of the required process element. For example, the process pattern in Figure 3b can be used to define a *technical review task* of a development process. This type of problems is ordinary but frequent.

The question for the second problem is *how to organize a group of process elements* to satisfy a special situation or certain constraints. Such problems focus on how to achieve high quality process models and effective executions of those models. A pattern

dealing with this type of problems provides a model for structuring a (fragment of) process, i.e. for establishing certain relationships between process elements. The pattern shown in Figure 2a, for example, provides a process flow that takes into account late information by allowing cloning process steps. Although this type of problems is currently overlooked in process patterns formalization, it exists and was discussed in many works [21, 22, 29].

During modeling a process, there are several scenarios where process designers may want to reuse process patterns. Based on the two above types of process modeling problems, we identified two main uses of process patterns as follows.

3.1 Using Process Patterns to generate the content of a process element

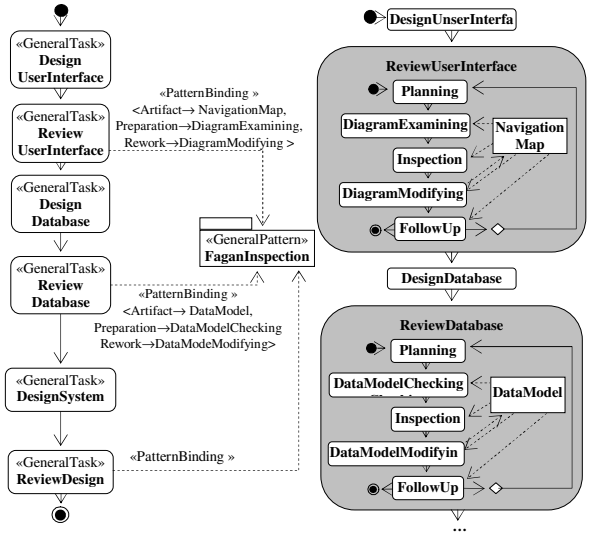
Often, in the process definition phase, process designers need to elaborate a detail description for a process element. In such cases, process patterns can be used as building blocks to construct new processes quickly. This is the most popular use, and to our knowledge, it is the unique use of process patterns proposed by the process community for process modeling.

In our meta-model, this kind of process pattern reuse is represented by the relation *PatternBinding* between a process element (source) and a process pattern (target). The presence of a *PatternBinding* relation implies that the contents of the model captured in the target process pattern will be copied into the bound process element.

To promote the use of process pattern, we permit the reuse by abstraction mechanism. We define process patterns as parameterized templates and allow explicit parameter substitutions in the relation *PatternBinding* to refine process models captured in process patterns on instantiating them[1].We show in Figure 4 an example illustrating the use of process patterns to generate process elements' contents through the relation *PatternBinding*. For the sake of simplicity, we won't represent the products in this example.

Figure 4a shows the baseline development process model of NSU containing unspecified process elements. When refining this baseline process, the process designer decided to apply the well-known Fagan process to review software artifacts. So he reused the process pattern *FaganInspectionPattern* (Figure 3b) to generate the content of the tasks *ReviewUserInterface*, *ReviewDatabase* and *ReviewDesign*. Because *FaganInspection* is a general pattern, for each application of it to generate a specific task, the designer specified the appropriate parameter for the binding. For example, the general task *Preparation* in the pattern was substituted by *DiagramExamining* for the bound task *ReviewUserInterface*; and by *DataModelChecking* in the bound task *ReviexDatabase*.

The unfolded models for the tasks *ReviewUserInterface* and *ReviexDatabase* are shown in Figure 4b.



(4a) NSU Baseline Process Model with unspecified process elements

(4b) Resulting model of the binding relation from *ReviewUserInterface and ReviewDatabase* to *FaganInspection* pattern

Figure 4 An example of process pattern application for generating a process element's content

3.2 Using Process Patterns to enrich or (re)organize a group of existing process elements

For process elements organizing problems, another way to apply process patterns is required. The general situation in such cases is that process designers want to apply new relations on a group of existing process elements, or to add new information to that group. Process patterns for such problems can be used as process structure templates to (re)organize processes. However, this kind of process pattern reuse has not been exploited yet in practical process modeling. We define therefore the relation *PatternApplying* to enable process structure template reuse[2].

A *PatternApplying* relation represents the application of a source pattern to a specific situation involving specific target process elements playing the roles of the elements defined in the pattern. The

[1] Due to the space limit, we cannot present here the detailed definition of this relation.

[2] As for the relation *PatternBinding*, we allow parameter substitutions in this relation. The extraction of our meta-model that defines these relations will be present in another paper.

presence of a *PatternApplying* relation implies that the structure of the process model captured in a source pattern will be applied to the target model. The specific way this application is realized depends on the *applyingMode* specified in the relation.

We define three *applyingModes* for the relation *PatternApplying*: *change*, *replace* and *extend*.

If the mode c*hange* is chosen, all relations (if existed) between target model's elements will be removed, only new relations and dependencies (together with associated elements) defined among the elements of the source pattern will be applied to the corresponding elements in the target model.

If the mode *replace* is chosen, the content of the source pattern will override the existing one of the target model. More precisely, in the case the target model contains elements and relations other than the source pattern, these existing elements will be conserved in the resulting model only if they are not contrary to the ones defined in the source pattern, otherwise they will be replaced by the pattern's elements.

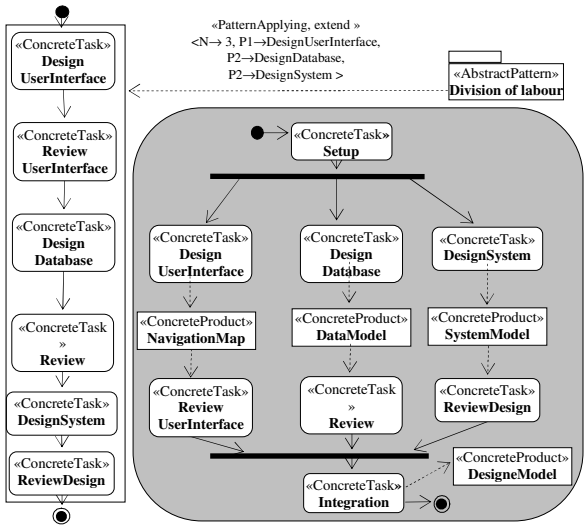
The mode *extend* indicates that the content of the source pattern will be merged with the existing one of the target model. However, the existing elements are intact and the new ones will be ignored if there are conflicts [3] between them.

To illustrate the use of process patterns to reorganize the structure of process models, we take a scenario happened at a NSU development team.

To design an information system, normally this team uses the concrete process described in Figure 3c. This process was constructed for NSU small teams that composed of one designer and one tester. Thus, the three components of design model, i.e. interface, database and system designs are elaborated and verified sequentially (Figure 5a). However, for a new project, this NSU team grew with three designers and three testers. To optimize the development time, the project leader wanted to change the design process to allow parallel works.

To satisfy this requirement, the process designer wanted to apply the process pattern *Division of labour* (Figure 3a) to reorganize the workflow of *NSU Information Systems Design process* but keep the contents of its elements which were fully specified.

Figure 5 shows this application through the use of *PatternApplying* relation.



(4a) NSU Baseline Process Model with unspecified process elements

(4b) Resulting model of the binding relation from *ReviewUserInterface and ReviewDatabase* to *FaganInspection* pattern

Figure 5 An example of process pattern application for reorganizing process elements

4. Conclusion

We presented in this paper a solution to broaden the definition and the use of process pattern for exploiting better this concept in process modeling. This proposition is formalized in a process meta-model that defines rigorously the process pattern concept and two operators for process patterns reuse.

Compared with these related works, our approach emphasizes process patterns reuse by abstraction mechanism. By distinguishing process elements at different abstraction levels, our meta-model allows representing various kinds of process knowledge encapsulated in process models. Based on this multi-levels abstraction, our process pattern definition covers a large variety of process patterns. By using parameterization to define process pattern concept and its reuse operators, we promote the use of process patterns and enable the pattern-based process model representation. Especially, the two proposed operators, binding and applying, provide the complementary ways to exploit process patterns for different process modeling needs, which are ignored in previous works.
However, the proposed formalism just allows representing the process knowledge that can be expressed as a set of process elements with relations among them. Further studies on potential conflicts in applying process pattern(s) to a target model should be investigated to enable a more rigorous semantic of this operator.

We are now attempting to formalize the actions to unfold the resulting process models of the proposed

[3] Conflicts may come from contrary dependencies between tasks' execution orders, between products' impacts; or from the incoherent relations between tasks and manipulated products, between tasks and participants roles, etc. The detailed analysis for such conflicts will be present in another work.

operators with OCL queries. Besides, a meta-process defining a systematic method for modeling processes based on patterns is currently developing. Finally, to validate our work, we are implementing a process modeling tool that supports the proposed meta-model and meta-process and allows process constructing and tailoring by reuse process patterns. A potential integration or alignment of this tool to the Eclipse Process Framework [4] is aimed.

6. References

1. Ambler, S. W.: Process Patterns: Building Large-Scale Systems Using Object Technology. New York - SIGS Books/Cambridge University Press (1998)
2. Basili, Victor R. and Rombach, H. Dieter.Support for comprehensive reuse. *IEEE SoftwareEngineering Journal*, 6(5):303–316, September (1991)
3. Derniame, JC., Kaba, BA., Graham Wastell, D. (Eds.):Software Process: Principles, Methodology Technology. LNCS1500, Springer-Verlag, (1999)
4. Eclipse Process Framework (EPF). http://www.eclipse.org/epf/
5. Fagan, M.E.: "Advances in Software Inspections". IEEE Transactions on Software Engineering, Vol. SE-12, No. 7, Page 744-751 (1986)
6. Firesmith, D., Henderson-Sellers, B.: The OPEN Process Framework. An Introduction. Addison-Wesley (2001)
7. Franch, X., Ribó, J. M. A UML-Based Approach to Enhance Reuse within Process Technology. EWSPT03
8. Gnatz, M., et al., Proceedings of the 1st Workshop on Software Development Process Patterns (SDPP'02), OOPSLA 2002, Washington. (2002)
9. Gnatz, M. Marschall, F., Popp, G., Rausch, A., Schwerin, W.: The Living Software Development Process. Journal Software Quality Professional, Volume 5, Issue 3 (2003)
10. Hagen, M., Gruhn, V.: Process Patterns - a Means to Describe Processes in a Flexible Way. ProSim04.
11. Henninger, S.: An Environment for Reusing Software Processes, International Conference on Software Reuse, (1998) 103-112
12. Hollenbach, C., Frakes, W.: Software Process Reuse in an Industrial Setting. *Fourth International Conference on Software Reuse*, Orlando, FL, (1996)
13. Jacobson, I., Booch, G. and Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley (1999)
14. Jørgensen, H. D. Software Process Model Reuse and Learning, Process Support for Distributed Team-based Software Development (PDTSD'00) Florida, (2000).
15. Kellner, M. Reuse of Process Elements. 10th Int'l Software Process Workshop. France.(1996)
16. Lonchamp J., Process Model Patterns for Collaborative Work, Telecoop'98, Austria. 1998
17. OMG: Software Process Engineering Metamodel 1.1, 2005
18. OMG: SPEM 2.0 Draft Adopted Specification, 2006
19. OMG, UML 2.0 Infrastructure Specification, 2005
20. OMG, Meta Object Facility Core Specification 2.0, 2006
21. Pavlov, V. and Malenko, D.: Mining MSF for Process Patterns: a SPEM-based Approach, Viking PLoP 2004, Uppsala , (2004)
22. Penker, M., Eriksson, H.E.: Business Modeling With UML: Business Patterns at Work. John Wiley & Sons (2000)
23. Perry, D. E. Practical Issues in Process Reuse. 10th Int'l. Software Process Workshop. France. 1996)
24. Ribó, J. M and Franch, X.: Supporting Process Reuse in PROMENADE. Research Report LSI-02-14-R, Dept. LSI, Politechnical University of Catalonia (2002)
25. Storrle, H.: Describing Process Patterns with UML, *EWSPT2001* (2001)
26. Tran, D.T, Tran, H.N., Coulette B., Crégut, X., Dong T.B.T. Topological properties for characterizing well-formedness of Process Components. Software Process: Improvement and Practice, Wiley Interscience, V.10 N. 2, p. 217-247, mai 2005
27. Tran, H.N., Coulette B., Dong T.B.T: A classification of Process Patterns. SWDC-REK 2005. Reykjavik, Island (2005)
28. Tran, H.N., Coulette B., Dong T.B.T: A UML based process meta-model integrating a rigorous process patterns definition. PROFES 2006, Amsterdam, (2006)
29. Van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros. A.P.: Workflow Patterns. Distributed and Parallel Databases, 14(3), pages 5-51, July (2003)