Table of Contents

Special Issue on Parallel Object-Oriented Programming

Guest Editors: Boleslaw K. Szymanski, Charles D. Norton

M. Austern, R. Towle, A. Stepanov: Range Partition Adaptors: A Mechanism for Parellelizing STL

S. Baden: Software Infrastructure for Non-Uniform Scientific Computations on Parallel Processors

D. Gannon, S. Diwan, E. Johnson: HPC++ and the Europa Call Reification Model

Y. Ishikawa: MPC++ Approach to Parallel Computing Environment

A. Chien: ICC++ - A C++ Dialect for High Performance Parallel Computing

C. Kesselman: High Performance Parallel and Distributed Computation in Compositional C++

C. Norton, V. Decyk, B. Szymanski: On Parallel Object Oriented Programming in Fortran 90
From Process Modeling to Domain Modeling

Luigi Benedicenti‡, Giancarlo Succi‡, Tullio Vernazza‡

‡ Software Production Engineering Lab (LIPS), Department of Communication, Computer and System Sciences, University of Genova, via Opera Pia 13, I-16145, Genova, Italy
e-mail: Luigi.Benedicenti@dist.unige.it, Tullio.Vernazza@dist.unige.it

† Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive N.W., Calgary, AB, T2N 1N4
e-mail: Giancarlo.Succi@enel.ucalgary.ca

Keywords
Process Modeling, Pattern Mining, Metrics, Software Reuse

Abstract
The hype on software process and software reuse as the capitalization on software as a valuable asset is encountering more and more appreciation. This paper presents an experience in modeling and extracting valuable process assets to build a framework in a small Italian firms specialized in telecommunications personalized solutions.
The company has been modeled at first by means of interviews and then analyzing the components in terms of fours main entities. Then, domain analysis was performed. The results of the two phases gave birth to a set of organizational patterns of reuse in the firm that lead to a general organizational framework. The framework can then be refined in a more specific meta-pattern definition or a design framework.

Introduction
Modern software development companies are seeking to be efficient in the software development process, to increase the value to the products they deliver, and to capitalize on the production assets. This is possible only if the asset capitalization goes up to the production process. This allows to understand, streamline, and reuse the process itself, making it a valuable good that can be traded internally or externally, subject to improvement and reengineering [1].

This paper presents the author’s experience in modeling a small Italian software company and translating the process knowledge in domain knowledge to extract the organizational patterns that compose its design framework.

The company we analyzed is a small (5 employees) Italian software producer, specialized in personalized high-end telecommunication solutions. The company prefers to remain anonymous.

Modeling their software development process took approximately one month, but the actual modeling went on for about a year to trace process changes. In this paper we first show the company's process model, then we describe the domain modeling process and finally we present an example of the organizational patterns derived.

**Process Modeling**

The process modeling toolkit we adopted is called Pescarenico [2]. It was initially developed as a result of a Process Improvement Experiment financed by European Union. It is a employs process modeling, process measurements, and process evaluations in order to obtain a clear picture of the process in the firm. It is supported by an automated modeling tool.

![Figure 1: Modeling Entities](image)

The modeling technique uses the OMT notation to draw a formal picture of the model in terms of four basic entities: people, roles, activities, and resources (Figure 1) [3], [4]. The process measurements are performed by the process actors themselves, by means of forms produced using a variation of the Activity Based Costing process accounting technique; they are used to "tune" the process model and produce a more detailed view of the ongoing process [5]. The process evaluation employs the balanced scorecards to produce a "dashboard" for the process, a collection of useful measures to track process execution and proceed with optimizations or reengineering [6].

The coordination of the three phases produces a process model that is fully integrated in the firm, and is easily understandable by anyone, facilitating any possible technology transfer.

The process model is therefore built interactively. It is a continuously changing element, and depicts the actual instance of the software development process in the firm. The modelers can keep track of the process history simply by storing the different process versions (shown in the diagrams). The modeling procedure started with interviews. The interviews obtained were used to characterize the process model, and draw a snapshot of the process itself. Figure 2 presents the snapshot.

The snapshot gives a good overview of the enterprise. At the beginning of the modeling process we realized that the roles and the main activities were already known to the firm's personnel, and that there was a good understanding of the specific involvement of each person. In fact, the model validation phase did not produce any high-level change. This is common in small firms where activities are not spread into functions, since there is only a single controller who is also the firm's CEO.

The "organization and control" activity shows the fairly centralized decision structure in the company. However, the "problem solving" activity involves everyone, which hints at some sort of delegation of the responsibility, typical of some small enterprises where agility and flexibility is a primary asset.
This initial scheme captures the organization’s structure and is useful to infer the organizational patterns that the firm has. Once these patterns are extracted, it is possible to complete them with domain information and design the high-level framework.

![Diagram](image)

**Figure 2: Process Snapshot**

It is important to point out that the interviews contain much more than is depicted in this initial model. However, some of the information can not go in the process model, and without a description of the application domain, is lost forever. This is why we proceeded with the analysis of the domain, and tried to correlate the two modeling methods.

**Domain Modeling and framework extraction**

The domain modeling phase started with the interviews. In fact, the interviews already contained information on the firm’s application domain. We extracted the information on the domain and we complemented the process model obtained in the first phase to help determining the organizational patterns that compose the high-level framework for the enterprise. To analyze the domain we applied the initial phases of the Proteus domain analysis method [7].

The organizational patterns we sought to obtain are high level patterns of organization. We adopted the pattern language defined by Coplien [8]. Although the level of patterns we use is very high, we believe that this level s the basis for defining a more operational level, exactly in the same way as the analysis phase of a software development process is followed by the design phase.

We obtained the following kind of candidates for the framework:

1) Patterns derived exclusively from the process model
2) Patterns derived exclusively from the domain analysis
3) Patterns resulting from the union of domain information and process information
Figure 3 shows an example of pattern derivation with three patterns, one for each of the cases above. The patterns themselves are characterized in Table 1. A good reference for them is in Coplien’s work [8].

![Diagram of Solo Virtuoso and De-Couple Stages]

Figure 3: Pattern Identification

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Extracted From</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solo Virtuoso</td>
<td>Domain</td>
<td>This pattern was found in the specific domain of the company. It derives from the way the software is produced in the company, and from the small size of the software code required by each product: the product can then be delegated to a single person who is held responsible for it.</td>
</tr>
<tr>
<td>Size the Organization</td>
<td>Domain + Process</td>
<td>The hint for this pattern came from the process model. The accounting and problem solving activities had sub-activities linked to evaluate and acquire new resources. This directly influences the size of the organization. However, the existence of the pattern was derived by the environment, and thus the domain requisites.</td>
</tr>
<tr>
<td>De-Couple Stages</td>
<td>Process</td>
<td>This pattern was inferred directly from the process.</td>
</tr>
</tbody>
</table>
model. The de-coupling of tasks are in fact a specific feature of the process, the way the roles and people are arranged, and their responsibilities.

After characterizing the organizational patterns, the modeling procedure can proceed further in the definition of meta-patterns [9], [10] and of design patterns [11].

**Conclusions**

This paper presented our work in modeling the process of a software company and conjugating it with domain information to obtain the basic organizational framework which is the starting point for a design framework.

Process modeling is extremely helpful in finding organizational patterns that domain analysis might miss. It is not an “orthogonal” procedure, in that it may overlap the findings of domain analysis. However, we found that it may prove useful to use both process modeling and domain analysis, to obtain a more precise organizational framework. In the future, we will formalize the union of the two techniques in a joint methodology.

**References**


