A software product line approach to enhance a meta-scheduler middleware

To cite this article: Rafael F Scheidt et al 2012 J. Phys.: Conf. Ser. 341 012030

View the article online for updates and enhancements.

Related content
- Modeling a distributed environment for a petroleum reservoir engineering application with software product line
  Rafael de Faria Scheidt, Patricia Vilain and M A R Dantas
- Towards a Resource Reservation Approach for an Opportunistic Computing Environment
  Eliza Gomes and M A R Dantas
- On the Optimization of GLite-Based Job Submission
  Giuseppe Misurelli, Francesco Palmieri, Silvio Pardi et al.
A software product line approach to enhance a meta-scheduler middleware

Rafael F Scheidt, Katreen Schmidt, Gabriel M Pessoa, Matheus A Viera and Mario Dantas
University of Santa Catarina (UFSC), Department of Informatics and Statistic (INE), 88040-900, Florianopolis - SC, Brazil
E-mail: {rfscheidt, katreenschmidt, gabrielm, matviera, mario}@inf.ufsc.br

Abstract. Software Projects in general tend to get more software reuse and componentization in order to reduce time, cost and new products resources. The need for techniques and tools to organize projects of higher quality in less time is one of the greatest challenges of Software Engineering. The Software Product Line is proposed to organize and systematically assist the development of new products in series at the same domain. In this context, this paper is proposed to apply the Software Product Line approach in Distributed Computing Environments. In projects that involve Distributed Environments, each version of the same product can generate repeatedly the same artifacts in a product that evolves its characteristics; however there is a principal architecture with variations of components. The goal of the proposed approach is to analyze the actual process and propose a new approach to develop new projects reusing the whole architecture, components and documents, starting with a solid base and creating new products focusing in new functionalities. We expect that with the application of this approach give support to the development of projects in Distributed Computing Environment.

1. Introduction
A Software Product Line represents a set of systems sharing common functionalities and characteristics that satisfies the needs of a particular market or specific segment [1]. Such a system set also can be named as Product Family [2]. The members of the product family are specific products systematically developed by the Software Product Line also classified as core assets. The core assets are represented by a variable features set which indicate a late decision of design of project [1]. The assets choice and configuration composes a specific product [3].

The Computational Grid approach in general presents a complex and distributed structure involving hardware and software that allows access with low costs to the high performance computing resources [4]. In the context of this approach, could find several elements like graphic interfaces to access the environment, middleware, meta-scheduler and so on, in consequence involve several knowledge areas that can derive other set of elements.

In this sense it becomes a challenge to systematically organize through the Software Product Line, such elements and architecture, aimed at planning systematic reuse and fast customization for products development that complete other needs and have the same structure.

This paper is organized as follows: Section II represents the value of concept Software Product Line. Section III describes the Distributed Computing Environments. Section IV illustrates and focuses
on the proposal and related works. Section V discusses the facts and problems found at the moment. Section VI provides comments and directions about future works.

2. Software Product Line

Software Product Line respects to systematical planning and strategic reuse productivity, by exploiting similar features of a product. Its main goal is to achieve significant reduction in organizations in terms of development, cost reduction in maintenance, reduce time-to-market, improve quality productivity and customer satisfaction as well to anticipate the problems encountered [1].

The main difference between conventional Software Engineering and Software Product Line is the variation presence in some or all the software requirements and the focus. In the conventional systems the focuses is a unique goal, in other words, produce a specific product. While the conventional systems development works ad-hoc, in other words, contract oriented, the Product Line development has a strategic vision of the niche market [5].

The implementation of Software Product Line approach requires efforts on the major artifacts in the development cycle. Table 1 shows the main characteristics of the essential artifacts for Software Product Line.

<table>
<thead>
<tr>
<th>Core Artifacts</th>
<th>Implementation cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Must support variation inherited of Product Line</td>
</tr>
<tr>
<td>Software Components</td>
<td>It should be design to be general, without loss of performance, must have support of variation points</td>
</tr>
<tr>
<td>Test Plans, Test Cases and Test Data</td>
<td>Should be considered variation points and multiple instances of Product Line</td>
</tr>
<tr>
<td>Business Case and Market Analysis</td>
<td>Should approach a family of software products, not just a product</td>
</tr>
<tr>
<td>Project Plans</td>
<td>It should be generic or extensible to accommodate variations of the product</td>
</tr>
<tr>
<td>Tools and Processes</td>
<td>Should be more robust</td>
</tr>
<tr>
<td>People, Skills and Training</td>
<td>Should involve training and experience about the artifacts and procedures associated with the Product Line</td>
</tr>
</tbody>
</table>

There are essential activities for Software Product Line. Such activities are Domain Engineering, which aims to develop the core of artifacts and Application Engineering, which will be instantiated in the core of artifacts to generate different products also known as Product Development [1].

Core assets are a set of assets ready to be reused in new product development. The core assets can be software components, project patterns, documents used in development, architecture, schedule and other artifacts that will serve as building blocks in the Product Line. The core assets are present in every product line. The architecture is one of these core assets and carries with it the possibilities of line variability. Architecture core asset, in particular, is seen as a key point of a product line and if it is projected badly can derail the entire project [2].

The main objective of Product Development is the generation of products based on core assets of a Product Line. It corresponds to the instantiation of applications and since found new requirements that had not been specified, it starts a feedback between this activity and the activity of core assets (Domain Engineering), in cycles in an evolutionary way.

A very important concept intrinsic to a Software Product Line is the variability. To form a software family through Software Product Line it is necessary identify possible changes that may occur in the
artifacts produced. The products generated during products development phase can be differentiated in terms of behavior, quality attributes, physical configurations, scaling factors, among others [2]. Variations are the tangible differences between the products of Product Line in any device such as architecture, components, interfaces between components and components connection. The variations can be identified in any development phase of Products Line [7].

To represent the possible variations identified in a particular domain becomes necessary an approach of variability, which until now, there is no consensus about how to identify and represent variability in Product Line because it is a recent issue [8]. One possible approach available and based on UML was developed and named SMarty (Stereotype-based Management of Variability) [8]. The SMarty approach was developed based on the activities and on the concepts of variability. A SMarty approach was created to manage the variability consisting of a UML profile called SMartyProfile and a process named SMartyProcess. The main goal of SMarty approach is to allow the variability of a Product Line to be managed effectively supporting UML models.

3. Distributed Computing

Grid computing is characterized by making a variety of distributed resources, including services, devices and applications, available to a wide range of users [9]. Various organizations, both real and virtual (VO), can make different types of resources available under dynamically changing availability constraints and with varying policies for access and use of these resources [10].

As a result of their nature, this variety of services and resources can be incorporated for achieving computational tasks. Specifically, users can access resources, applications and services, submit jobs for execution either via queues or by advance reservation, create combination processes in workflows, and verify the status of jobs or systems. Also, there has been a movement towards integrating grid computing environments with mobile computing [11] [12], creating an interface for users access the resources and services of a grid from anywhere, at anytime.

One the other hand, clusters environments are common configurations used in many organizations to reach high performance computing (HPC) for specific local applications. Multi-clusters, if well orchestrate as a grid environment, can represent a differential computational power for a global execution of several class of applications, including also parallel jobs. A grid environment, composed by multi-clusters configurations, can be considered in a private or in multi-domain organizations. In other words, this metacomputing [13] environment can be employed as a HPC facility inside a specific organization, or among different organizations, creating in both cases the concept of virtual organization (VO).

One important challenge is how to reach an efficient re-utilization of this environment and coordinate all resources and services.

4. Proposal and Related Works

With a focus on Distributed Computing Environment the present study aims to systematically organize the elements and architecture, aiming the systematic reuse planning and rapid customization for new product development to complete other requirements and to have the same structure. To achieve this objective, this work applies the SMarty process specifically in the project being developed by the laboratory LaPesD at the Federal University of Santa Catarina UFSC, contributing to the development of techniques to systematize a process for reuse of the artifacts, architecture and components of Software Products Line in Distributed Computing Environment.

Among the existing approaches of Software Product Line and researches, SMarty approach was chosen due to its ability of modeling artifacts and ability of integrate with the UML.

Currently in the literature are few works in the area of Distributed Environments. One of the few existing, but with excellent quality [14] [15] proposes to build an approach of Software Product Line for Grid-Oriented Middleware. The work developed a Software Product Line Architecture to facilitate
the development of Grid-Oriented Middleware Systems. In order to validate the proposed PLA, it is instantiated in the construction of a middleware for grid computing.

Another approach is named GISPL (Grid Infrastructure Software Product Line) adopts SPL concepts in the design, optimization and implementation of Grid-Oriented Middleware. Initially, the requirements are defined through ontology and a specific language. Next, service oriented architecture is defined that incorporates software and hardware artifacts needed to the functioning of VO (Virtual Organization). The architecture is modeled in System Modeling Language (SysML), which allows specifying the infrastructure using specific models derived from the ontology defined in the previous step. Finally, the infrastructure models are prototyped and optimized [16] [15].

Unlike the presented approaches, the purpose of this study is to apply in the Distributed Computing Environment as a whole, covering all the part of domain and instantiating complete applications distributed at the stage of product development.

5. Discussion
The initial motivation to use a Software Product Line approach in the distributed environment, which is often a complex architecture, was based on problems encountered, such as: architecture elements are deployed and are removed a few times, because the client does not know exactly what you want about a project. Such problem without using Software Product Line causes a massive rework.

In the proposed architecture, each element presented (GUI Interface, Meta-scheduler, schedulers) can be componentized as part of a common architecture called Domain Engineering (DE), and then every new product would made an analyze to an instance of this Architecture of Application Engineering (AE) and specialization of the parties not common.

![Diagram of Domain Engineering in Distributed Computing Environment](image)

**Figure 1.** Domain Engineering in the Distributed Computing Environment.

For example it can apply ontologies aimed to describe the existing settings in the available network resources specializing only one of the architecture components. It can also add new elements or components in the architecture, such as an upper layer to replace the GUI for a mobile application with...
an ontology itself (where this component would be part of the Domain and later reused in other products).

All these elements presented in the architecture and in any layer can be planned carefully developed thinking of reuse and variations points (mandatory, optional and alternative) aimed at creating a Software Product Line in the Distributed Computing Environment.

Study the possibility of using OSGI [17] on the GUI layer and on the cellular phone. This technology increases the level of componentization lowering the granularity of each module.

6. Conclusion and Future Works

This paper presented an approach to Software Product Line architecture specially designed for distributed computing environments. The principal relevance for this approach can be defined as follow.

Increasingly through the Software Engineering seeks to reuse elements for reuse in project from the same domain. This practice is directly related to improvement in the cost of projects, time and productivity [6]. The Software Product Line contributes with all these features and to increase the quality of final product, productivity and so on.

The adoption of Software Products Line concepts provides a way for better integration of environment components quickly and securely, and also supports an approach of mass customization.

The reuse of the artifacts and final product decreases the number of errors, increases compliance of the produced software, eases maintenance and reduces development time.

We are now applying the approach in the projects and documenting information about this. With the application of the approach, we are able to solve practical problems related to the user’s indecision as to what elements in its architecture the user would like to have and quickly add and remove elements without major problems.

7. References


