The importance of Documentation, Design and Reuse in Risk Management for SPL

Luanna Lopes Lobato  
UFPE – CIn - Recife - PE, Brazil  
Tel: +55 81 2126-8430  
UFG – DCC - Catalão - GO, Brazil  
Tel: +55 64 3441-1510  
lil@cin.ufpe.br

Pádraig O’Leary  
UFBA – DCC - Salvador – BA, Brazil  
Tel: +55 81 3283-6258  
padraig@dcc.ufba.br

Eduardo Santana de Almeida  
UFBA – DCC - Salvador – BA, Brazil  
Tel: +55 81 3283-6258  
esa@dcc.ufba.br

Sílvio Romero de Lemos Meira  
UFPE – Cln - Recife - PE, Brazil  
Tel: +55 81 2126-8430  
C.E.S.A.R - Recife – PE, Brazil  
Tel: +55 81 3425.4700  
srlm@cin.ufpe.br

ABSTRACT
Software Product Lines (SPL) is a methodology focusing on systematic software reuse, multiple benefits have been reported as a result of this type of software development. However, establishing a SPL is not a simple task. It is a challenging activity raising many challenges for engineering and management. This research aims to manage the risks during SPL development to provide traceability among them. For this, it is important that the risks are documented and there is a common design related to them. As solution, we identified the strengths and weakness in SPL development and the importance in designing of communication for risk documentation.

Categories and Subject Descriptors
D.3.3 [Software Engineering]: Management.

General Terms
Documentation, Design, Standardization, Verification.

Keywords
Software Product Line, Documentation, Web Products.

1. INTRODUCTION
A Software Product Lines (SPL) is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [1]. It involves the management of variabilities and commonalities among several applications, which increases its complexity compared to Single System Development (SSD). Thus, developing a SPL requires time and systematic planning to achieve positive results; otherwise the investment can be lost due to failures in the project.

According to Boehm and DeMarco (1997) software developments risky nature is easy enough to acknowledge in the abstract, but harder to acknowledge in real-world situations. Therefore, it is necessary to predict, control and manage risk.

Documents can be used for risk avoidance; their value should be evident for a project manager. Since team members will almost always face similar obstacles, it is imperative that they share how they overcame these obstacles. Documentation can ensure the same mistakes are not repeated in the next project. This is particularly true for SPL development as several products are typically derived from the same platform.

Documentation plays an important role in the capturing of tacit knowledge particularly when there is a high rate of staff turnover. Thus, it is important to capture the knowledge about the project. Therefore, if a team member moves on, the valuable knowledge, good practices and lessons learnt will not be lost.

Risk Management (RM) documentation is important for avoiding recreating the risks already identified. Insights into RM documentation were identified in a systematic literature review in RM for SPL and through two SPL projects, where the risks were anticipated and resolved. Thus, we have identified risk and the mitigation strategies to avoid them.

This research aims to manage risk within SPL and define how these risks can be represented, providing design of communication among them. The risks were mapped in documents and designed in a way to provide insights about them through cross-referencing. This enables traceability amongst risk. Reuse across RM is focused on the ability of reusing this documentation for different products across the SPL. Therefore it is necessary to verify the lessons learnt from previous experiences and to predict, control and manage risk.
It is also important to highlight that RM documentation is a ‘live’ document, as in it should be constantly (re-)visited and updated during the project lifetime and after its conclusion. This is in order to gather ongoing and postmortem information about the project, mainly about new risks that were not previously identified.

Thus, this work is intended to analyze and document the risks inherent in SPL development using the literature and case study research to identify risks in software projects. This documentation is important to provide information about the magnitude, impacts probabilities and mitigation strategies about the risks.

RM in SPL is an open issue with currently little research in this area.

The methodology that will be used, to provide the design of communication for the risks identified, will be best practice research on SPL, through the findings provided by academy and industry. This information will be used to collect insights about RM in SPL development.

The remainder of this paper is organized as it follows. A brief overview of SPL and RM is described in the next subsection. Section 3 carries a discussion based on state of the art, highlight the related works. The research scenario is described in Section 4, emphasizing the risks and lessons learned observed, which will be used to provide design of communication on risks identified. And, we present the conclusions and some promising venues for future research in documentation for RM in SPL.

2. SPL WITH RISK MANAGEMENT

Software Product Lines (SPL) is an approach for software development focusing on systematic reuse. Benefits regarding the effort reduction, time-to-market and quality improvement in software development have been reported as a result of adopting SPL through its systematic exploitation of reuse opportunities.

According to Schmid (2002) these benefits (effort reduction, quality improvement and time-to-market decrease) arise through reuse opportunities that are presented due to the similarities between systems within the product line. SPL involves the management of different products features, where variabilities and commonalities are managed among several applications.

However, establishing a SPL is not a simple matter. The approach is particularly complex, raising specific challenges for engineering and management. SPL adoption involves major investment and considerable risks are associated with it. Moreover, technical and non-technical aspects have to be addressed to obtain the benefits of organization-wide software reuse. In this context, it is necessary to manage the risks that may be presented during the SPL development, in a way to improve development and provide the project success.

SPL adopters are typically more concerned about the issues related to the technical aspects of the development, such as, architecture development or domain analysis (Schmid, 2001). However, just technical capabilities for an SPL cannot guarantee the adoption success. The development must be supported by an auxiliary method that helps the stakeholders to make decisions during the development process. Thus, other aspects must also be relevant to manage the risks present in SPL development and provide a design of communication about them.

RM supports SPL development in order to provide process improvement and problem avoidance within the project. For RM additional investments are necessary to start the development focusing on quality services. Consequently, RM practice will avoid some problems that occur in some stages of SPL development, based on the management of the actions that must be following and performed during the SPL introduction.

Boehm (1989) defines RM as an attempt to formalize risk into a readily applicable set of principles and practices. The goal is to identify, address, and eliminate software risk items before they become either threats to successful software operation or major sources of software rework.

Simply stated, risk is the possibility of loss or damage [6]. This definition can be translated into the fundamental concept of RM: “risk exposure” (also called, “risk impact” or “risk factor”), likelihood, contingency plan and mitigation methods. Implementing RM involves inserting the RM principles and practices into existing life-cycle management practices. The key contribution of software risk management is to create this focus on critical success factors - and to provide the techniques that let the project deal with them.

In large project development, the product lines benefits are stressed and very little attention is paid to possible risks [7]. However, in practice, it is important to notice and understand which risks can happen, if they may turn into problems or opportunities and how they should be dealt with.

The literature reports that RM can significantly improve software project outcomes [8]. There are many risks in software projects, from several sources, which need to be controlled during the software development process.

Thus, RM is particularly important in development projects due to the inherent uncertainties that most software projects face. It is necessary to apply risk-reduction activities. These activities are designed to minimize a particular risk or group of risks i.e. to minimize the likelihood that a problem corresponding to the risk will occur [9].

In the article by Hartmann (2006), according to CHAOS 2004 report, in SSD, only 29% of the projects were finished successfully; 18% of the projects failed without giving any delivery and 53% of the projects were finished with overtime or over their budget. The report states that the causes of failure in software development projects are also related to failure in RM. A software development project is rich in strategic opportunities, but it is also subject to multiple sources and high levels of uncertainty. When managed properly, uncertainty creates opportunities, thus it is important to manage the uncertainty [11].

Despite of the problems found with risk occurrence in projects, risk itself is not bad, it is essential to improve the project and failure is often a key part of learning. However, according Van Scoy (1992), we must learn to balance the possible negative consequences of risk against the potential benefits of its associated opportunity.
RM it is not just a process for avoiding risk. The aim of risk management is not to eliminate risk, rather to manage the risks involved in all activities to maximize opportunities and minimize adverse effects. More specifically, risk management is a formal (business) process used to identify risks and opportunities across the organization, assess the potential impact of these events and then provide a method for addressing these impacts to either reduce threats to an acceptable level or achieve opportunities.

This paper aims to emphasize what are the most common risks and provide adequate mechanisms or techniques to mitigate them, explaining how they can be handled or even avoided. Risk documentation is necessary to avoid the same problems that can become real in projects. With RM it is possible to provide the design of communication, with goals for establish the traceability among the risks.

As supported for this method, we verified the documentation about decisions, functionalities, that were present in SPL and SSD in the literature, and lessons learned. Thus, the documentation and design of the risks are reused in several products to avoid the same problems faced during software projects development.

Despite evident benefits in adopting RM during software project development, it is not an easy task to define and to use guidelines to capture and share what really worked and did not work in a project. It is necessary to establish the culture inside the team members. Therefore the design of communication, through documentation and reuse of assets, are important points to provide this management.

3. STATE OF THE ART

To identify the SPL risks reported by the literature a systematic review was performed. Through the systematic review, it was possible to collect the most important characteristics to provide design of communication in relation to the risks in SPL development. In addition, we identified what were the main research gaps facing RM in SPL.

According to our systematic analysis, we verified that no approaches have been developed for RM in an SPL context and, when some studies exist, they are defined without details. Therefore, although studies have shown that RM is necessary to achieve software project success, RM still faces obstacles before being institutionalized by SPL companies.

In this research, the risks in software development are identified with an emphasis on the RM in SPL. With the lack of an existing approach for RM in SPL, this work has been developed. It stressed the importance of documenting the risks identified and the reuse the strategies defined in the documentation to resolve the risks in future projects.

Therefore, the design of communication of the risks in SPL is the main focus of this work. Communication of risks is essential to provide insights about the traceability among them. This research comes to facilitate the projects development, providing a documented list about the risks, which must be monitoring throughout the SPL development. With this strategy of risk documentation, managers should be able to identify, prioritize, and quantify the risk types involved in their projects.

3.1 Related Works

According to Bobun, Pozgaj and Sertic (2003) the essence of risk classification is not in precisely defining risk categories, but in identifying and describing as much risk as possible in the project.

This work intends to identify a set of best practice to build a more reliable and effective approach for RM in SPL through the use and reuse of documentation. In our research, we highlighted the importance of risks documentation and design, as well as the reuse of risk documentation, emphasizing the need to document risks in relation to design of communication.

Throughout our systematic review in RM in SPL, we observed an increase in the number of published papers covering RM in SPL over the last 10 years. This recent increase may be a reflection of a growing awareness of the importance of motivating RM for SPL. Alternatively, this increase may just match a general rise in published papers in SPL. Researchers and practitioners have to rely on practitioner books in order to get information on RM approaches or, at worst, not apply any strategy to control risks in SPL development. This means not having sufficient contingency plans and relevant mitigation plans to provide for RM.

In a systematic review of RM in SPL, it was observed that risk strategies and mitigation techniques are not addressed by all literature that mentions them. The handling of these remaining risks is actually an open question. Some risks are described, in a superficial way.

Despite the benefits in SPL, there are high risk level due to its inherent complexity. Thus, SPL development is more complex and demanding than SSD.

The follow research is the most important for our context. Initially, Schmid (2002) propose an approach related to a reuse process with the development of PULSE-ECO. Next, Voget and Becker (2002) describe some risks for immature scope, highlight that it is a risk if the scope is vague, is subject to changes or the scope size is inadequate. In 2003, Riva and Del Rosso showed research described some of risks that are identified in SPL approaches, such as: Missed Schedule, Malpractice in Management, Failure in Requirements Identification, Core Assets Instability and Slower Process of Change. Four years later, the concept of product line assessment is proposed by Olumofin and Mišić (2007).

The literature on RM for SPL presents a conflicting and partial scenario. It is clear that there is a need for RM practices in SPL development, however, there are few studies which present relevant information. Most studies mention the importance of RM, although the steps for managing risks – as well as their results – are not clearly specified.

Highlighting this figure, it was possible to observe that there are needs for extensive research in this area. Further empirical studies should be performed with sufficient rigor to enhance the body of evidence in RM within SPL engineering. In this context, there is a clear need for conducting studies comparing alternative methods. In order to address scalability and popularization of the approaches, future research should be invested in risk support and in combined SPL adoption strategies, mainly in documentation, design and reuse of the products assets.
4. RISKS AND LESSONS LEARNT

In this section we will present the lessons learned during our research. Not much data has been published on experiences and lessons learned for RM in SPL in a way that demonstrates the design for communication of those risks. In this section, we show practices that can be used during the SPL development as strategies for avoiding risk.

These insights were collect through a systematic literature review, in an industrial and academic context. Thus, we collected and documented feedback and lessons learnt from various studies, which we can use to demonstrated the direction of RM within SPL.

In the following section, we show these risks and what are the lessons learnt that can be linked to resolve them. These are documented with goals to provide insights for the manager. The risks are identified by ID and name and classified according to type.

4.1 Project Risks

R1-Complexity of SPL:

Product line architectures pose unique challenges, which are not present in single product architectures. This difference makes the assessment of such architectures rather difficult. At the most superficial level, quality attribute scenarios for SPL are more numerous than in the case of a single product architecture, SPL architecture is more context dependent, and they are more complex as well [16].

Architectural issues are also a few of the many issues that have to be considered. SPL approach involves more than architectural issues. If no due allowance is made for these multiple aspects, the chances of success will be small. This helped us compare business, architectural, process and organizational aspects of the approaches.

Compared to developing a SSD, an SPL requires extra resources in process and tools development, core development, and product management [17].

R2-Inadequate CM:

A configuration management plan, which could integrate smoothly and systematically with the configuration management system of individual projects. We believe that the shared responsibility for the maintenance of the product line assets will result in great benefit, as any software product can take advantage of any problem fixing or in-service experience of the rest of the software products [26].

R3-Malpractice in Management:

SPL demands software management and development practices, which are capable of coping with new levels of organizational and architectural complexity [27].

Experiences shows the importance of synchronizing needs, defining roles, communication between core asset team and implementation team for architectural integrity, and using proper tools for dependency analysis [18].

Management at the technical (or project) and organizational (or enterprise) levels must be strongly committed to the SPL effort for the product line’s success [23].

R4-Inadequate Technical Documentation: As scoping starts early in the SPL development life-cycle, little documentation exists and the exchange of information needs to rely on personal communication. Existing approaches are usually not sufficiently documented to be replicable, nor is validation data available that allows assessing the contribution of the approach to the field [3]. The documentation became too bulky or the product developers could not find the right information [7]. Poor description of the generic architecture and requirements documentation is a risk for project success [27]. The lack of documentation on implicit properties frequently results in mistakes. Similarly, in some projects if there insufficient, irrelevant and voluminous documentation [31].

R5-Lack of SPL Information:

While commercial tool vendors do not yet pay particular attention to SPL-specific needs, a few research prototypes from universities show that SPL support is possible [27].

R6-Absence of Domain Experts:

With the growing investment by both public and private sector organizations in software product-line architecture, managers need to know the organizational factors affecting its success [25].

R7-Missed Schedule:

From an economic viewpoint, a leveraged SPL adoption entails a revolution, because the company can address a completely new market segment with low costs and few risks by building on an existing PL infrastructure. Different technical domains, even within the same SPL, can vary considerably in terms of their potential benefit and inherent risks for product line engineering [19].

R8-No product focus:

The main goal of the product family architecture is to describe the commonality and variability of the family in order to make explicit the variation points of the products [15]. Product line requirements are traceable to higher level specifications (e.g., stakeholder requirements in different projects), and also to product line analysis- and design artifacts (e.g., use cases, state machines and components) [22].

R9-Immature Architecture:

The assessment of the SPL architectures has been based on methods developed for single product architectures. The complexity of evaluating PL architecture like the dual form of the architectures, existence of variation points, the need for associating context with the large number of quality attributes scenarios and the need to perform quality tradeoff analysis across the SPL architectures has largely been ignored [16].

Technical factors do not by themselves explain the success of product-line architecture. Only in conjunction with appropriate organizational behaviors can software architecture effectively control project complexity [25].

R10-Pollution of the platform:

Technically excellent product-line architectures do fail, often because they are not effectively used. Some are developed but never used; others lose value as product teams stop sharing the common architecture; still others achieve initial success but fail to keep up with a rapidly growing product mix. Sometimes the
architecture deterioration is not noticed at first, masked by what appears to be a productivity increase [25].

R11-Platform not Mutable:
Introduction of new requirements in a dynamic market is critical to handle the forthcoming requirements in time. Even though the problem of incorporating new requirements is not specific to product family architectures, the process has to accomplish an even more difficult task [15].

R12-Core Assets Instability:
When the business becomes more mature: New investments are needed to consolidate the software assets. The various sets of products are migrated towards a SPL in order to keep all the software variants under control. The organization needs to adjust its operating procedures to support the global management of the products lifecycle (from requirements engineering to testing) [15].

R13-Inappropriate Reuse Activity:
Reuse may decrease system integrity as the potential failures of the new system might have not been considered in the original analysis of the reused software. The configuration management of the product line has therefore two dimensions: Managing the artefacts ‘across time’ (version update) and ‘across different projects’ [26].

Reuse has always been considered the main approach to achieve major improvements in productivity and quality in software engineering. Consequently, much work, both from academia and industry, has been undertaken with reuse as target [30].

R14-Slower process of change:
The evolution of a SPL is mainly driven by two forces: the consolidation of the assets in the platform and the creation of new products. The platform slowly evolves by incorporating the new architectural requirements, while new products are added by introducing new features [15].

R15-Lack of Tool Support:
Establishing a traceable link between the requirements, design, and the reusable code can enable the deployment of reusable software in an effective and analyzable way. Therefore, reuse should be addressed holistically, rather than being limited to the code, where most reuse attempts have failed [26].

4.2 Process Risks
R16-Immature Process:
A SPL increases quality, shortens time to market, and helps specify the “right” product features. Strong process structures help manage complexity and conflicts between the requirements of individual products and projects [17].

R17-Immature Domain:
Product lines are only successful if the underlying domain releases arrive in due time with necessary quality level and the major functional contents. From a software engineering perspective this is an obvious contradiction. Impacts cannot be fully assessed and micro-managed in advance, neither can risks or people skills [24].

4.3 Product Risks
R18-Inadequate Quality of the Artifacts:
Successful product line engineering requires management and coordination of both the core asset and product development projects to meet the organization’s overall business goals [28].

This implies the use of repeatable and systematic procedures to ensure that the set of requirements obtained is complete, consistent, easy to understand and analyzable by the different actors involved in the development of the system [29].

R19-Excessive Variability:
The variability of the products must be considered when evolving the architecture and it must be carefully verified if a requirement for a product can lead to a break from the product family architecture. In the analysis of the forthcoming requirements it must be ascertained how easy it is to add them to the current architecture and estimates the work needed for the implementation [15].

Proper variability management is one of the key success factors of a software product family. In practice, however, variability management suffers from a number of issues that prevent organizations from exploiting the full benefits of software product families [31].

R20-Inadequate Features Definition:
A commonly used product line requirements modeling technique is feature modeling. One of the strengths of feature models is that they provide a good and easy way to get an overview of the common and variable parts of a product line [22].

R21-Failure in Requirements Identification:
Requirements Engineering and Management (REM) is a central task of product line development. The challenges in REM occur, mainly, because SPL development is much more complex than SSD and SPL requirements management is considerably harder than requirements for single-product development [27].

4.4 Staff Risks
R22-Inadequate Communication:
Due the SPL complexity, several insights about the project development can be lost. Thus, the use of questionnaire can help during the identification of the problems. It is crucial to make the assessment results independent of the particular persons conducting the evaluation and to make the assessments repeatable over time (in the sense, that if you ask twice, you get two times the same – or at least a similar answer– if circumstances did not change in the meantime) [20].

R23-Problem with Staff:
Project and technical managers within an organization need to be assured that the reusable assets of a product line are reliable and trustworthy, particularly when project teams do not have full control over the development of these assets [26].

Product line business practices cannot be affected without explicit management commitment and involvement. Many product line efforts fail for lack of sponsorship and commitment from someone above the technical ranks [23].

R24-Shortage of skilled labor:
Skilled architects can mitigate challenges in architectural development [17].

4.5 Organization Risks

R4-Difficult to introduce a SPL:

In practice it is relatively difficult to introduce a product family approach. Many initiatives do not achieve their goals or even fail because their impact and potential problems were not properly identified [7].

Building a SPL is a long-term effort in which the benefits come through reuse, which can only come after several product releases. According Jaaksi (2002) building a few products first is the right way to initiate a product line [17].

Whenever an organization is embarking on a large scale reuse approach, like introducing PL engineering, this entails a serious investment and it is important to pose the question: ‘what are the potential benefits we will be able to reap and what will be the risks that we might encounter?’ [20].

Although adopting SPL can be beneficial, there is a perception that it has high barriers to entry. One barrier is the perceived prerequisite of a SPL architecture [21].

SPL development requires an organizational mind shift. When moving from SSD to SPL, several related products must be envisioned together to develop an architecture/design that can fulfill the requirements for an entire family of products [22].

The business and process contexts require the transition to be incremental, and the architecture therefore needs to support this through explicit definition of implementation proposals [18].

The community needs more quantitative data to support SPL adoption. Moving to product lines is an investment, and decision makers want hard numbers in their business cases [23].

Many industries are hampered with introducing the product line concept into already existing products. Though appealing, the concept is very difficult to introduce specifically into a legacy environment. All too often the impacts and risks are not considered adequately [24].

R25-Non-use of certifications:

SPL artifacts should ideally be certified once and reused in different products without further certification effort. However, in practice, the certification authorities consider only completed systems [26].

4.6 Business Risks

R26-Limited development costs:

The product line increased costs in various support functions, architectural work, and management. Thus, it is suggested that an organization considers initiating a product line only when it both aims at systematic reuse and serves a heterogeneous customer base with a common domain [17].

R27-Delay in Time-to-Market:

Technical, business and environment requirements change at a tremendous speed. The ability to launch new products and services with major enhancements within short timeframe has become essential for companies to keep up with new business opportunities. The need for differentiation in the marketplace, with short time-to-market as part of the need, has put critical demands on the effectiveness of software reuse. In this context, SPL approach has become one of the most established strategies for achieving large-scale software reuse and ensuring rapid development of new products [18].

5. THE IMPORTANCE OF DESIGN IN COMMUNICATION TO RM IN SPL

Lack of documentation is a problem during both single system and SPL development. It is important to explicitly define the documentation about the domain, scope, platform, assets, product architecture and configuration management. This was re-enforced by: [2], [14], [16], [23], [24], [26], [27]. It is important to explicitly define the documentation for the domain, platform and product architecture, defining a meta model for requirements documentation and ensuring the document quality [27].

Documentation is as important as any other aspect of the project, but the problem is that we do not realize its importance. Documentation should be overriding to all projects, however few mention it, just 8 studies, of the 30 analyzed related this issue: [3], [14], [16], [23], [24], [26], [27].

According to these 8 studies documentation plays an important role. It is essential to document the scoping decisions to identify the mistakes and document the scoping definition process [3]; Thus, it is emphasize that the decisions must be documented [16]; It is important to define documentation about the domain, platform and product architecture explicitly [27]; Thus, it is possible to keep configuration management plans actualized [26]; For this, special emphasis is put onto the documentation of the potential commonalities in the work products [14]; During management the operations and the SPL efforts communication paths are documented in an operational concept [23]; the requirements must be documented in a structured and disciplined way [24].

In additionally, interviews with stakeholders are important to collect insights about the possible problems that can occur. Several studies mentioned about this strategy: [3], [16], [25], [27], [20], [22], [29].

In conclusion, according to our systematic analysis conducted, it was verified that, in general, few approaches have been developed to deal with RM in the SPL context and, when some study exists, it is defined briefly. Although studies have shown that RM is necessary to achieve success in software projects, this area still faces obstacles before being institutionalized by companies working on SPL projects. Additionally, the design of communication about the risks in SPL still needs more extensible research.

6. CONCLUSIONS

This work involves the investigation and integration of two research areas: Software Product Lines and Software Risk Management, and the needs to design the communication between them. After the general introduction to the problem and motivation for RM in SPL, we analyze the issues surrounding RM in SPL identify what needs to be accomplished in order to build a useful body of knowledge.
RM is a practice that deserves greater attention during SPL development, however, there is a lack of research in this area. This conclusion is confirmed by Birk and Heller (2007), where complexity also has a particular impact on requirement engineering management. These challenges occur, mainly, because of the greater complexity in SPL: technical issues (e.g., very high number of features, feature interaction, interrelations between architecture and requirements) as well as organizational and managerial issues (e.g., very many stakeholders, many interrelated projects and releases). Also, for most of the SPL specific issues, tool support is widely lacking.

As briefly conclusion of the risks and lessons learnt identified in the literature were presents, highlight the needs in provide documentation about them. Most approaches do not address the evaluation of risks that are connected with the specific ways to document product line development. Those approaches that address this do it in at a high level. This leads only to rather coarse-grained evaluations [3].

In this context, there is a clear need for conducting studies comparing alternative methods. However, according to Kimer, and Concalves (2006) despite the studies and experiences published about risk management, the software industry, in general, does not seem to follow a model to analyze and control the risks through the development of their products [32].

This work intended to identify risks in the literature and report the importance in use this risks during the project development. As solution can be establish a set of good practices to build a reliable and better effective documentation and design of the risks identified, and reuse the strategies for avoid them.

To the best of our knowledge our research is the first one to aim to support explicitly RM in SPL, focusing on a characterization of the risks involved through the whole SPL process and as well as the methods documented to treat them. With the results, we can suggest an approach that deals with RM and tackles the risks present in the software development process.

The successful adoption of SPL engineering requires a profound organizational mind shift. The whole software engineering process is affected from requirements to maintenance and evolution activities [33]. However, this continuous management is not described in research studies.

We believe that this research will be useful for both, academy and industry. The research clarifies the importance of documenting risk and to capture and reuse past project experiences. The risks in SPL are supported by scientific studies, providing some directions on which aspects to focus. This highlights the importance of providing documentation about projects, and to reuse this documentation in a way that avoids recurrence of the same problems. These results will help in decisions about the development of an approach to RM in SPL, with experience about what are the strategies and steps that will be useful, as well as, what are the risks of major impacts.

Future research is to identify which are the common risks between SPL and SSD and, these results give us basis to propose the approach to RM associated with SPL development.

7. ACKNOWLEDGMENTS
We would like to thank at Informatics Center in Federal University of Pernambuco. In addition, our acknowledgments to CNPq. This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

8. REFERENCES

1 http://www.ines.org.br


