Available online at www.sciencedirect.com
ScienceDirect
The Journal of Systems and Software 81 (2008) 996–1013
The Journal of Systems and Software
www.elsevier.com/locate/jss

Software reuse: The Brazilian industry scenario

Daniel Lucrédio [a,*], Kellyton dos Santos Brito [b,c,1], Alexandre Alvaro [b,c,1], Vinicius Cardoso Garcia [b,c,1], Eduardo Santana de Almeida [b,c,1], Renata Pontin de Mattos Fortes [a], Silvio Lemos Meira [b,c,1]

[a] *ICMC – Institute of Mathematical and Computer Sciences – University of São Paulo (USP) – Av. Trabalhador São-carlense, 400, Centro 13.560-970 São Carlos/SP, Brazil*
[b] *Informatics Center – Federal University of Pernambuco (UFPE) – Av. Professor Luis Freira, S/N, Cidade Universitária 50.740-540 Recife/PE, Brazil*
[c] *Recife Center for Advanced Studies and Systems (CESAR) – Rua Bione, 220, Cais do Apolo, Recife Antigo 50.030-390 Recife/PE, Brazil*

Received 31 May 2007; received in revised form 20 August 2007; accepted 23 August 2007
Available online 14 September 2007

Abstract

This paper aims at identifying some of the key factors in adopting an organization-wide software reuse program. The factors are derived from practical experience reported by industry professionals, through a survey involving 57 Brazilian small, medium and large software organizations. Some of them produce software with commonality between applications, and have mature processes, while others successfully achieved reuse through isolated, ad hoc efforts. The paper compiles the answers from the survey participants, showing which factors were more associated with reuse success. Based on this relationship, a guide is presented, pointing out which factors should be more strongly considered by small, medium and large organizations attempting to establish a reuse program.

Keywords: Survey; Software reuse; Reuse success factors; Survey; Best practices

1. Introduction

Software reuse is the process of creating software systems from existing software rather than building everything from scratch (Krueger, 1992). Systematic reuse is domain focused (Frakes and Isoda, 1994), based on a repeatable process and includes a concern for high-level artifacts, such as requirements and design models (Frakes and Isoda, 1994). Its potential effect on competitiveness is profound (Frakes and Isoda, 1994; Frakes and Fox, 1995), since organizations compete within domains, and the one that better understands a particular domain(s) will have competitive advantage.

The reuse community initially concentrated its research on technical issues, such as repositories (Mili et al., 1997, 1998; Seacord, 1999; Guo and Luqi, 2000; Burégio et al., 2007), search-based tools (Lucrédio et al., 2004) and domain-specific languages (Deursen et al., 2000). However, as more experience became available (Endres, 1993; Griss, 1995; Joos, 1994; Rothenberger et al., 2003; Frakes and Kang, 2005; Sherif et al., 2006), non-technical issues, such as organizational structures, processes and human involvement, appeared to be at least as important (Morisio et al., 2002).

Many organizations are implementing systematic reuse programs and need answers to practical questions, both technical and non-technical, such as:

- What kind of development should be adopted? Component-based? Software product-lines? Aspect-oriented development?

* Corresponding author.
E-mail addresses: lucredio@icmc.usp.br (D. Lucrédio), kellyton.brito@cesar.org.br (K. dos Santos Brito), alexandre.alvaro@cesar.org.br (A. Alvaro), vinicius.garcia@cesar.org.br (V.C. Garcia), eduardo.almeida@cesar.org.br (E.S. de Almeida), renata@icmc.usp.br (R.P. de Mattos Fortes), silvio@cesar.org.br (S.L. Meira).
[1] Tel.: +55 81 3425 4700; fax: +55 81 3425 4701.

0164-1212/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2007.08.036

- How the development team should be organized and trained?
- Which languages and tools are better suited for reuse?
- Can reuse be achieved in any application domain?
- Is my organization too small to try establishing a reuse program?
- What are the steps for introducing reuse in my organization?

Even with some experiences from industry, we need more empirical data which analyzes the advantages and drawbacks of introducing reuse in software organizations. However, little empirical data is available to help answering these and other questions (Frakes and Fox, 1995; Morisio et al., 2002; Rine, 1997). There are many successful stories available on the literature, but these can not be easily generalized to other contexts (Almeida et al., 2005; Bayer et al., 1999).

Besides, it is very difficult to determine exactly what makes a reuse program successful, mainly because it depends on the organization structure, background, budget, among other technical and non-technical factors. Each organization has its own characteristics, such as size, experience, application domain and available resources, for example.

In this sense, this paper presents the results of a survey that attempts to relate the characteristics of software development organizations with successful reuse adoption, in order to determine which development factors, under which circumstances, have more influence on software reuse success. Twenty-one factors were considered, divided into four perspectives: organizational factors, business factors, technological factors and processes factors. Each perspective groups related factors, and one perspective is somehow isolated from the others. This division is useful for organizations, which can put special focus on different parts of reuse adoption, one perspective at a time. Another possibility is to assign one manager for each perspective, according to his/her experience or technical background, so that each group of factors may be dealt with simultaneously by specialized professionals.

One particularly important factor is the organization size. Small organizations sometimes wonder whether systematic reuse is a realistic goal, given the scope of their domains and limits on their resources. Large organizations sometimes feel that instituting systematic reuse is unrealistic because of the large investments of resources and time required (Frakes and Fox, 1995). In this sense, we also present, in the end of the paper, a guide for small, medium and large organizations, showing which factors are more important in each case, according to the results of the survey. Although the guide can not be considered as the final word on the subject, it compiles important and recent experiences related by industry professionals that already obtained success (or not) in reuse adoption in their organizations, pointing out which factors should be more carefully analyzed when attempting to establish a successful reuse program.

Although the survey involved Brazilian organizations only, the results can be applied for other countries as well. We enforce this fact by showing, in Section 4, a comparison with other important studies involving organizations in other countries, including the US and some countries from Europe. Although these related studies did not analyze the exact same factors, the ones in common have similar results, which not only reinforces the findings of these related studies, but also increases the confidence in the findings presented here.

The remainder of this paper is organized as follows: Section 2 presents the research approach, including the planning, data collection and analysis phases, as well as a discussion on the study's validity. Section 3 presents the survey results, discussing the main factors for introducing reuse in software organizations. Section 4 presents the related works. Section 5 presents a guide for small, medium and large organizations, describing which factors are more important in each case. Finally, Section 6 presents the concluding remarks and future work.

2. Research approach

The approach for this research was systematically organized into three phases. In the *first phase*, the goal was to define, evaluate and validate the survey, and to define the target software organizations. In the *second phase*, the survey was sent to the software organizations, and the data were collected. In the *third phase*, the data was analyzed, with the objective of determining which factors have more influence on software reuse success. This section presents these steps, and discusses the main threats to the study's validity.

2.1. The survey

The survey consisted of a questionnaire developed after an extensive review of the literature, with a strong influence from previous works related to reuse surveys (Frakes and Fox, 1995; Morisio et al., 2002; Rine, 1997; Kitchenham et al., 2002; Clements and Northrop, 2002) and from industrial evidence experienced by our research group – the Reuse in Software Engineering (RiSE)[2] group (Almeida et al., 2004). The first version of the survey was defined in January 2004 and was revised for two months. The revision was accomplished together with statisticians, and with industry experts with several years of experience, including software developers, market specialists and TI managers. Most people involved in the process had been working in industrial software reuse projects for at least one year.

In order to evaluate the survey, a pilot project was made in conjunction with two software organizations. A set of non-technical improvements was made to increase the quality of the survey, such as including a privacy statement

[2] http://www.rise.com.br.

and definition of the target public. A statement about the estimated time for answering the questionnaire – 20 min – was also included in this phase.

2.1.1. The questions

The main goal of our study was to identify which factors are more important to reuse success, and which are less important. In this sense, the most relevant question asked to the participants was if they managed to achieve success in software reuse. Since software reuse is usually associated to productivity and quality (Rine, 1997), we asked if the software organizations obtained productivity and/or quality gains in some software development project, **as a result of performing some kind of software reuse**. It is important to stress that this is not an exact measure of reuse success, and therefore the results can not be treated as a formal and final evidence. But it allowed us to capture the industry professionals' impressions on software reuse, and to determine if some kind of benefit was obtained.

The other questions aimed at identifying possible factors that could have some influence on reuse success. After an extensive literature review (Almeida et al., 2005; Frakes and Fox, 1995; Morisio et al., 2002; Rine, 1997; Rine, 1997), and from our experience in reuse projects, we identified 21 factors related to software reuse, and important for organizations: *Software organization and team size*, *Project team experience*, *Software reuse education*, *Rewards and incentives*, *Independent reusable assets development team*, *Product family approach*, *Kind of software developed*, *Application domain*, *Software development approach*, *Programming language*, *Repository systems usage*, *CASE tools usage*, *Quality models usage*, *Systematic reuse process*, *Kind of reused assets*, *Origin of the reused assets*, *Previous development of reusable assets*, *Specific function in the software reuse process*, *Software reuse measurement*, *Software certification process*, *Configuration management of the reusable assets*.

These factors are divided into four perspectives: organizational factors, business factors, technological factors and processes factors. In order to identify which factors are present in the surveyed organizations, we included one or more questions for each factor. Some factors can be easily identified through direct questions, such as which kind of software is developed, or what is the organization's size. Other factors required more subjective data, and therefore some open-ended questions were included, such as:

```
Did your organization manage to obtain success (productivity/quality gains) in projects through software reuse?

(a) ( ) Yes. Which factors were responsible for the success?
          -
 ---------------------------------------------
(b) ( ) No. Which factors inhibited the success?
          -
 ---------------------------------------------

In your organization, do the assets pass through a certification process in order to assure its reusability?

(a) ( ) Yes. How is this process carried out:
          -
 ---------------------------------------------
(b) ( ) No.
```

The final questionnaire is composed of general information and a set of 26 questions, with 16 closed questions and 10 open questions, divided into three groups: (i) *general information*, such as geographic region, number of employees, number of employees involved with software development, among others; (ii) *information not directly related to reuse*, such as the software development team experience, programming languages used, software development approaches, software quality models adopted, among others; (iii) *information related to reuse*, such as success through software reuse, systematic reuse processes adoption, kind of reused assets, among others. The questionnaire is available online.[3]

2.2. Data collection

The questionnaire was sent to Brazilian industry professionals, during lectures and courses ministered by the group, where the participants were asked to fill the questionnaire, and through mailing lists. The questionnaire was sent to several important Brazilian software organizations, including the Digital Port[4] – a cluster composed of about 120 software organizations.

In some cases, there were more than one person from the same organization answering the questionnaire. In this survey we considered only one answer per organization, so we tried to merge the answers from the participants from the same organization, by comparing similar answers and adding complementary information from one questionnaire in the other. There was only a few cases where this happened, and we found no conflicting answers, so the merge could be accomplished without any trouble.

2.3. Data analysis

After the data was collected, we tried to analyze the different characteristics of the software organizations that could be the cause of the productivity and/or quality benefits associated with reuse success. The approach was to associate the percentage of claimed success in software reuse with each characteristic. We analyzed, for example, how reuse success is distributed according to the size of the organization, the characteristics of the development

[3] http://www.rise.com.br/research/Questionnaire.pdf.
[4] http://www.portodigital.org.br/.

team, in terms of size and experience, the kind of software being developed, the problem domain, among others.

Next, according to the percentage of success for these characteristics, we identified which ones are relevant factors in successful reuse adoption, and which are not, through a descriptive analysis of the data. Different reuse factors behave significantly differently (Rine et al., 1998). While in some cases the reuse success can be clearly associated to some particular factor, in other cases a difference can only be seen while analyzing several data simultaneously. In order to better establish such relationship between the different factors and reuse success, and to identify possible explanations for the results, we observed the following indicators:

- *Percentage of reuse success*: If some factor has a strong influence on software reuse, the percentage of reuse success suffers great variation. For example, 71% of the organizations that adopt a quality model claimed to achieve success in reuse, while only 45% of the organizations that do not adopt a quality model claimed to achieve success. This is a clear indication that the adoption of a quality model has great influence on software reuse. For this analysis, we also observed possible outliers, and the number of respondents for each factor. For example, 100% of the organizations that use aspect-oriented development claimed to achieve success in reuse, but this corresponds to one organization only. Finally, in some cases the participants did not answer all the questions (some was left in blank). These were excluded from the calculation of the percentages.
- *Qualitative data*: The quantitative data were confronted with qualitative data obtained from the answers of the survey, and from our personal experience in developing industrial reuse projects. This was helpful to confirm some of the results of the percentage of reuse success, and to try to understand the cases where the percentage of success was not clear enough to derive some conclusion.

According to the descriptive analysis, we assigned four different levels of influence that a factor may have on reuse success: **strong**, when the observed data (both quantitative and qualitative) clearly indicate a relationship between that factor and reuse success, **weak**, when some relationship can be seen for some aspect, but not very clear, **none**, when no relationship can be seen, and **no data**, when further research is needed in order to determine if that factor has some influence on software reuse.

2.4. Validity

We discuss three kinds of validity: construct, internal and external.

2.4.1. Construct validity

Construct validity considers whether the metrics and models used in a study are a valid abstraction of the real world under study (Morisio et al., 2002). In our case, this refers to the quantitative and qualitative indicators used to establish the relationship between the factors (independent variables) and reuse success (dependent variable).

The most important indicator in our study is reuse success, which we define as described in the previous section. As pointed out by Rine et al. (1998), the software engineer's perception on his organization's reuse capability is often correct, and therefore we believe that the question about reuse success, together with the analysis described in Section 2.3, produced reliable results. Besides, our definition of reuse success was extensively discussed between the authors, and with other professionals, such as project managers, statisticians and software engineers, as described in Section 2.1, reducing the chances of not including some important consideration. Finally, the pilot project also helped to increase the confidence on this indicator.

The other indicators, related to the different factors, reflect simple situations and characteristics of software organizations, that were extracted through direct questions, and answered without problems by the participants. Due to this simplicity, we believe that they have little chance of being biased by personal interpretation or containing highly subjective information.

Also, as will be further discussed later, some questions failed to determine the exact relationship between some factors and reuse success, because in some of them the respondents could choose between multiple options, but there was only one question about reuse success. For example, regarding the application domain, several respondents selected more than one domain, and therefore we could not determine which domain was responsible for reuse success. We further discuss these problems in Section 3.

2.4.2. Internal validity

Internal validity considers whether the experimental design is able to support conclusions on causality or correlations (Morisio et al., 2002). The size of our data is too small to allow meaningful statistical studies, so we adopted a more descriptive analysis.

Another possible threat to this study's internal validity is that we analyzed each factor separately. Although it served to determine which factors should be more or less considered by organizations in the path towards reuse, the factors are not independent from each other, and a multivariate analysis would be interesting to determine further relationships (Rine et al., 1998).

2.5. External validity

As the participants were not randomly chosen, this study cannot be generalized to describe the entire Brazilian software organizations behavior. Also, different participants answered the questionnaire in different situations. While some respondents provided their answers right after a lecture, others sent the questionnaire by e-mail, having

more time to think about the questions. This could lead to some differences in the answers. However, due to the simplicity of the questions, and the short time required to fill the questionnaire (20 min), we believe that the impact of this threat was minimum.

3. Survey results: factors influencing reuse success

This section presents the analysis of the data collected in the survey, discussing the relationship between the different factors and the percentage of claimed reuse success. First, the overall results are presented, followed by the detailed discussion on the different factors influencing reuse, organized in four perspectives: organizational factors, business factors, technological factors and processes factors.

Because our sample is not too big, we also provide, in addition to the percentage of success in each factor, the actual numbers, so that the "real" information is not hidden behind the percentages.

3.1. Overall results

At the end, about 200 software organizations were contacted, and 57 (28.5%) answered the survey. The participant roles included: software managers, system analysts, system engineers and software developers. Fig. 1 shows the number of participants of the survey. By collecting answers from professionals with different backgrounds, we were able to analyze different perspectives, and how reuse involves technical and non-technical points of view.

53% of the software participants (30 participants) claimed to obtain success in software projects due to software reuse. The remaining 47% (27 participants) either tried to adopt software reuse without success or did not try any software reuse approach at all. The reason why these organizations did not obtained success in reuse was not considered in the survey, since the goal is to determine what could lead to reuse success, and not what factors could lead to failure. This limitation of our work can be further explored in future works, similarly to Morisio et al. (2002) and Frakes and Isoda (1994).

Nevertheless, some participants also pointed out, in the form of comments in the questionnaire, some factors that contributed to failure in software reuse adoption, such as: *the lack of a formal process or methodology for software reuse*; *and the lack of knowledge of the organization or the development team about software reuse*.

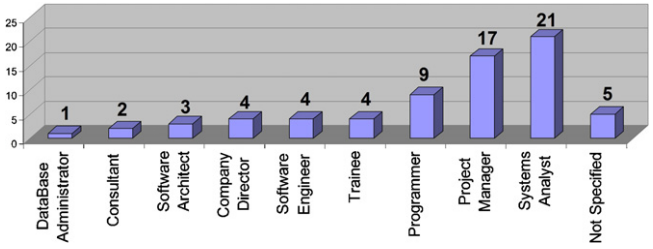

Fig. 1. Number of participants of the survey.

It must also be stressed that these claims on reuse success are based on subjective analysis, instead of measured results, since most software organizations do not have reuse metrics (see Section 3.5.7).

The software organizations that answered the survey were, mainly, located in Northeastern and Southeastern regions of Brazil, with a bigger sampling in the Northeastern region, where this study was mainly conducted, near Digital Port. Fig. 2 presents the relation between the software organizations location and reuse success collected in the survey. As it can be seen in the Figure, the percentage of organizations that claimed to achieve success in software reuse is exactly the same (58%) in the two main regions that were analyzed Northeastern (18 out of 31) and Southeastern (11 out of 19), which indicates that Brazilian organizations are in a similar level regarding software reuse. For the other regions, the amount of participants was too small to extract any meaningful conclusion.

3.2. Organizational factors

3.2.1. Software organizations and team size

In our experience with software reuse projects, we observed that small software organizations have software reuse as their goal, since the domain scope is more restricted and the resources are limited. Additionally, in these kinds of projects the manager is closer to the development team and, thus, has better control over each activity. Moreover, often, the team is smaller and the communication flows better. On the other hand, larger software organizations are normally afraid of deploying a software reuse program, mainly because they need a long-term investment and time to deploy the program effectively. Also, it is more difficult to introduce changes in large organizations, specially those related to software reuse, which include, among others, changes in the organization's culture and processes.

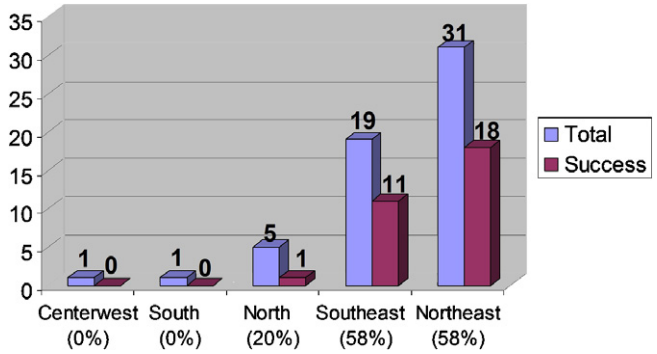

Fig. 2. Number of surveyed software organizations in each Brazilian region. The numbers between parenthesis indicate the percentage of organizations that claimed to achieve success in software reuse, for each region. (Obs: The same format of this figure is used in the next figures, so this explanation regarding the presentation of the percentage of success is suppressed from the remainder of the paper.)

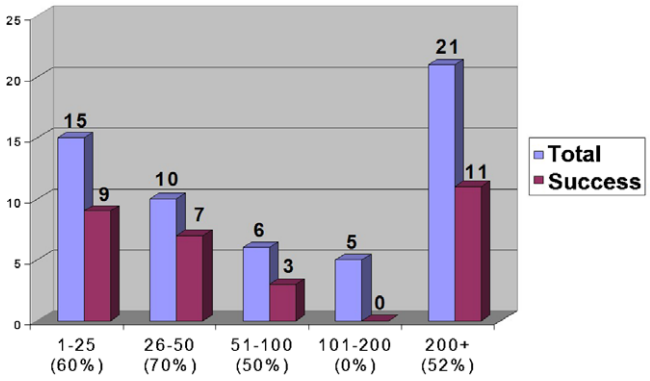

Fig. 3. Influence of organization size on reuse success.

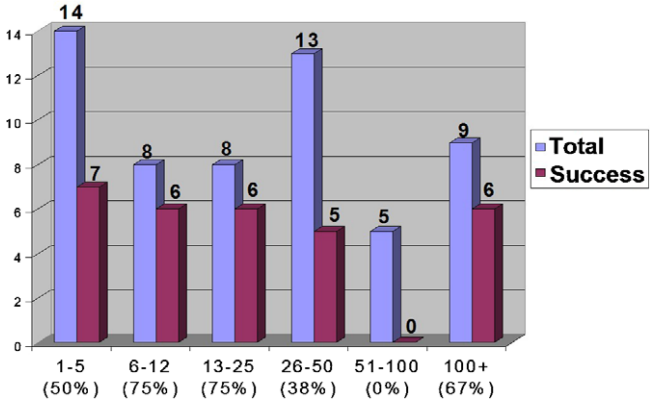

Fig. 4. Influence of the software development team size on reuse success.

This behavior can be observed in Figs. 3 and 4. Fig. 3 shows that the size of the software organizations is related to software reuse success. In the survey, smaller software organizations (1–50 employees) presented higher success rates than medium-sized software organizations (51–200 employees). The survey also showed that large software organizations (over 200 employees) reported 52% of success in reuse programs.

Fig. 3 considers the total number of employees of the organization. When considering only the number of people working with software development (Fig. 4), a similar pattern occurs, with smaller and larger teams having more success than medium teams.

Fig. 5 shows this pattern. In very small teams, reuse is achieved, but in a smaller percentage than medium teams. From this point, reuse success decreases as the number of people involved in the organization or the project increases, until a point where we observed no reuse success. In larger organizations, reuse success can be observed again.

A possible explanation for this behavior was obtained from the answers and from our experience in reuse projects. In smaller teams, reuse is more opportunistic and dependent on individual efforts, with people reusing software in an ad hoc way, by asking a colleague for some piece of code, for example. This is why very small teams

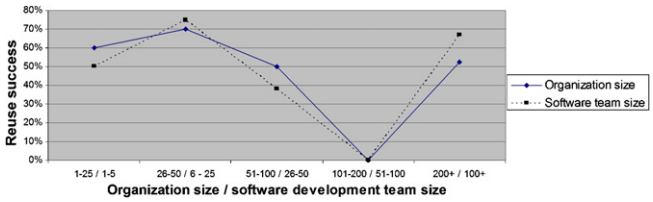

Fig. 5. Reuse success according to organization and software team sizes.

has a smaller percentage of reuse success, since in these cases the individuals end up with many simultaneous tasks, and so they are not able to spend time trying to promote reuse. On the other hand, in medium teams the individuals have more freedom to take the initiative of promoting reuse. For these cases, we observed 75% of reuse success.

But this kind of reuse becomes nearly impossible when the team size reaches some limit (which we observed as being around 50 developers, in an organization with around 100 employees), and a more controllable process is needed. In our survey, we identified that most of the large organizations that claimed to achieve success in reuse adopt a systematic software development processes. On the other hand, none of the surveyed medium-sized organizations adopted a systematic software development process. This explains why large organizations managed to achieve success in reuse, even with a large number of individuals, indicating that the existence of a systematic development process has great importance in reuse success (Rine, 1997), specially for large organizations. This was in fact observed in our survey, in the form of questions about quality models and systematic reuse processes, as will be further explained in the following sections.

Based on these observations, we conclude that organization and software team size, as an isolated factor, **does not** directly influence software reuse, since reuse success is possible in organizations of any size. However, according to the organization size, different factors should be analyzed. This reinforces the need for a guide for small, medium and large organizations trying to adopt reuse, presented in the end of this paper.

3.2.2. Project team experience

A common reasoning is that software developers with more experience would naturally obtain more success with software reuse, since their knowledge on some domain is bigger than the knowledge of less experienced teams. On the other hand, if the existing reusable assets have good documentation, where the knowledge and design rationale are properly registered, they are easier to understand and reuse (Paiva and Fortes, 2005), and so experience becomes less important.

Some researchers, such as Frakes and Fox (1995), state that experience does not help in promoting reuse, unless this experience comes from proper reuse education. At the time of their study (1995), reuse education was very uncommon (Frakes and Fox, 1995), and so they were

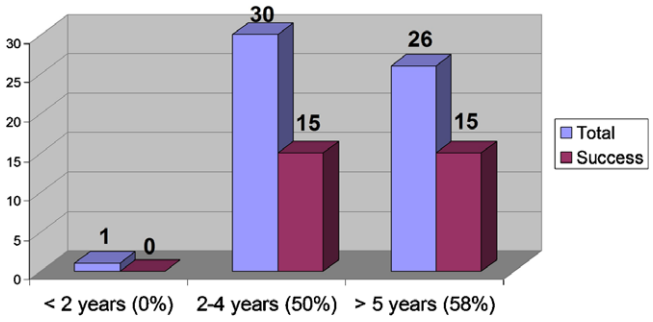

Fig. 6. Influence of developer experience on reuse success.

unable to find any correlation between experience and reuse success.

Fig. 6 summarizes our results. A short advantage was observed for the organizations that had professionals with more experience, and therefore we conclude that the team experience has a **weak** influence on software reuse success.

Additionally, as described in next section, our results show that reuse education in Brazil is still poor, both in the graduation courses and inside the companies, so the same explanation provided by Frakes and Fox (1995) may still be true.

3.2.3. Software reuse education

Usually, graduate courses do not offer specific disciplines that approach questions about software reuse. In fact, Frakes & Fox observed, in their research in the US (Frakes and Fox, 1995), that only 13% of the engineers involved in their study had education about software reuse in their institutions. To the other 87%, the only option is to do it alone or in the organizations they work in. Aiming to verify if Brazilian organizations are concerned with this education, we asked in our survey if they had a sector that is responsible for the introduction and maintenance of the conscience of software reuse, and if there were training programs focused on reuse.

Our study showed that, nowadays, few organizations adopt policies related to reuse education: only 15% (5 out of 33) of the organizations that answered this question had a reuse education program. Considering the total of software organizations that answered the survey, only 8% of them had this kind of program. Thus, our results confirm Frakes & Fox's observations, indicating that the situation has not changed much since 1995, and that reuse education is still not being properly addressed, at least in Brazil.

Regarding the influence of reuse education on reuse success, we verified that the existence of a sector responsible for the introduction and maintenance of the conscience of software reuse, for itself, has no influence on the degree of success in reuse programs, as well as the existence of training programs related to reuse. The rate of success for software organizations adopting these policies were 57% (4 out of 7) and 60% (3 out of 5), respectively, against 71% (39 out of 55) of success for the software organizations that did not adopt any reuse education program. Even comparing the results of the organizations that answered both questions affirmatively, the results were not expressive, with 58% (7 out of 12) of success.

In our experience we also observed similar results. We are working to define some tools and processes for reuse, and we are disseminating the reuse culture in our projects at C.E.S.A.R.[5] In general, we observed that the reuse level has not changed after this effort.

Therefore, we conclude that reuse education, as it is being currently carried out by the organizations, has **no influence** on software reuse.

3.2.4. Rewards and incentives

Reward policies have been presented as a stimulus to effective reuse adoption inside organizations, where the rewards are usually recognition or money. Some studies show results related to rewards. In some situations, it was verified that monetary rewards stimulate the software reuse and the success of reuse programs. On the other hand, Frakes and Fox (1995) did not collect data in organizations that used the monetary reward, and concluded that the recognition rewards did not have a significant contribution to reuse. Our study tried to verify, in the Brazilian scenario, what kind of rewards organizations were adopting, and if they contributed to reuse success.

Only 3.5% (2 out of 57) of the total of organizations answered this question affirmatively; however all of them are still in the initial phases of their reuse programs, without data to confirm if their programs had success and if the rewards contributed to it. After analyzing the comments in the questionnaire, we verified that the kinds of rewards being adopted are: employee's visibility inside the organization in events, dinners and money.

Therefore, we found **no data** that relates rewards and incentives with reuse success.

3.2.5. Independent reusable assets development team

Usually, organizations find themselves in a dilemma between the creation of an independent team for the development of reusable assets and the construction of the assets by the existent development teams. Creating an independent team can produce good results, because this team will be specialized in reuse, building more robust and well elaborated assets, minimizing the effort to reuse them. However, creating this team involves problems such as the high cost of their allocation and size, since it cannot be sub-used nor over-used (Lynex and Layzell, 1998; Fichman and Kemerer, 2001).

In the analysis of the data related to this question, we verified that few organizations had independent teams for developing reusable assets, around 9% (5 out of 57), but all of them (100 %) obtained success with software reuse,

[5] http://www.cesar.org.br/.

while 63% of the organizations that do not have an independent reusable assets development team reported reuse success. These data indicate that an independent team is a good practice, having a **strong** influence on reuse success.

3.3. Business factors

3.3.1. Product family approach

The product family and software product line approaches are different terms for the same concept (Pohl et al., 2005; Clements and Northrop, 2002), which can bring benefits to software reuse, mainly because the scope of the reusable assets must be defined in the early stages of development (John et al., 2006). Some authors, such as Frakes and Isoda (1994), Rine et al. (1998) and Morisio et al. (2002), based on practical experience, state that software organizations that develop systems based on a product family have increased probabilities of reusing software. Krueger (2006) states that the extended efficiency and effectiveness obtained with methods for product line development can go even further, opening a new realm of strategic and competitive advantages over conventional development.

This is due to the fact that product families have functionalities that can be shared. Besides, the experience with product families usually turns the development team into specialists in that domain, contributing to the success levels. On the other hand, software organizations that develop isolated products have more difficulties in selecting and searching their reusable assets. Moreover, they do not become specialists in any domain, because each project involves a different domain, and therefore reuse is more difficult to achieve. The successful stories in small, medium, and now large companies.

Our survey confirm these statements from the literature. 63% (17 out of 27) of the surveyed software organizations which were based on a product family approach claimed to obtain success in reuse, against 41% (12 out of 29) of success from others. Thus, we conclude that the software organizations that develop software based on a product family have a significant advantage in the reuse success levels, when compared to the software organizations that develop isolated products. Therefore, this factor has a **strong** influence on reuse success.

3.3.2. Kind of software developed

An important question about software reuse (but with little data available) is if the kind of software system that is developed influences software reuse. This study tried to identify this question, by mapping the software organizations through six categories of software: *Information Systems*, *Real-Time Systems*, *Embedded Systems*, *Intelligent Systems*, *Games* and *Others*.

According to the data collected by the survey, we verified that many software organizations develop Information Systems in conjunction with other kinds of software. In this way, the precise conclusion about the success levels in software reuse and the kind of the software developed becomes difficult. The reason for that is the fact that the survey did not allow to indicate in which kind of software the organizations obtained success, as explained in Section 2.4.1.

However, as can be seen in Fig. 7, we noticed that three kinds of software had the same percentage of reuse success (50%), while software organizations that developed Intelligent Systems and Embedded Systems had higher percentages of success than the others. Although this may indicate that some kinds of software are more amenable to reuse than others, most of the categories had a similar reuse success level. Therefore, we conclude that the kind of software developed has **no influence** on reuse success.

3.3.3. Application domain

In order to map in which domains it is easier to put in practice the reuse techniques, we suggested 13 possible application domains for the participant to choose. He could also write the domain his/her software organization works with, in the case it was not present in the list. The results are presented in Fig. 8.

We can observe in Fig. 8 that all software organizations that develop software tools (five organizations) had success with software reuse. A possible explanation, which we obtained after analyzing the other answers of these five organizations, is that since these companies focus on building software for developers, they have a larger concern with software engineering techniques and systematic processes, which seem to be a more important factor in the achievement of reuse. Besides, these correspond to only five software organizations, which is a small number compared to other domains, such as Administrative or Financial, so this result may not be considered very conclusive. We can also

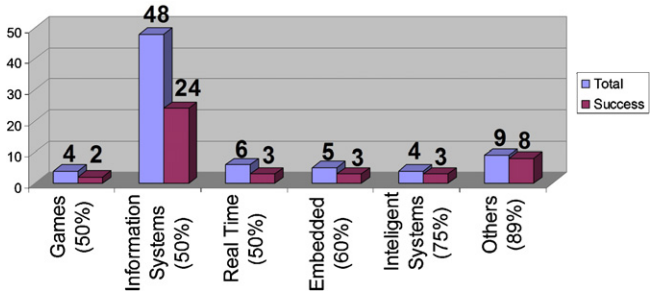

Fig. 7. Influence of the kind of software developed on reuse success.

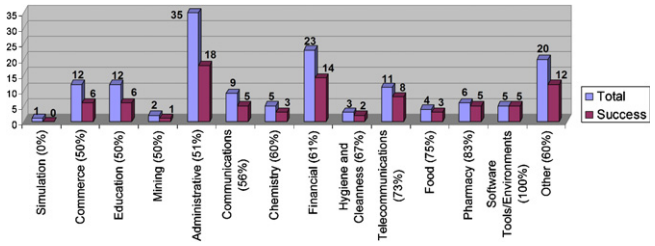

Fig. 8. Influence of the application domain on reuse success.

observe that the Simulation domain had no success, but that this corresponds to only one organization.

Excluding these extremes, we observed that most domains have around 50% and 60% of success, except for Drugstore, Foods & Drinks and Telecommunications, which presented a higher degree of success. In this sense, we conclude that there are only a few domains where reuse is more successful, and therefore this factor (application domain) has a **weak** influence on reuse success.

3.4. Technological factors

3.4.1. Software development approach

Nowadays, the software industry believes that object-oriented paradigm contributes to the success in software reuse, mainly, due to the concepts of inheritance, polymorphism and abstraction, support by this paradigm (Anastasopoulos and Gacek, 2001). Morisio et al. (2002) highlighted that the development approach and the programming language do not influence software reuse. In our study, we try to investigate this fact. Although some software organizations used more than one software development approach, it was possible to obtain some conclusions.

Only one software organization used aspect-oriented paradigm (AOP) in its projects. Although it obtained success with reuse, a single example is not enough to extract any conclusion regarding the influence of AOP on reuse. Except for that, the most successful levels occurred in software organizations that used the component-based development approach (64%, 7 out of 11), followed closely by the object-oriented approach (59%, 27 out of 46). The procedural paradigm had the worse success percentage (43%, 16 out of 37). These results are different from Frakes & Fox's statement, since the procedural paradigm may be introducing some kind of difficulty in software reuse success.

Additionally, correlating the success in reuse with the software development approach and team experience (Fig. 9), we can verify that the success level in software organizations that use procedural approach varies significantly when the team experiences change. This does not occur with the OO paradigm. This suggests that the success of the software organizations that use procedural language depends on the team experience, while with the OO paradigm the experience is less important.

Based on these observations, we conclude that the development approach has a **weak** influence on reuse.

3.4.2. Programming language

We also analyzed the impact of the programming language on reuse. The answers showed that all organizations use more than one programming language, and so we could not establish a direct association between the language and reuse.

However, we observed that the software organizations that use.NET (C#, ASP.NET, VB.NET) had a greater reuse success level than the ones using C++, ASP, C and JAVA. The result of.NET is expected, since.NET is not just a programming language, but is an integrated environment that brings some benefits to developers. ASP and C had better results than JAVA, contradicting the common belief and the results presented in the previous analysis, which pointed the object-oriented paradigm as an advantage in promoting reuse. However, this may be caused by the fact that software organizations use ASP together with the.NET platform, and so the success can not be attributed exclusively to ASP. The same effect occurs with Visual Basic. Fig. 10 summarizes these data.

Since the reuse success percentage does not vary strongly across the different languages, we conclude that the programming language has a **weak** influence on software reuse.

3.4.3. Repository systems usage

A reuse repository is a collection of reusable assets with requirements such as search and retrieval mechanisms (Mili et al., 1998). The use of a repository as an essential factor for reusability has been target for debates. Frakes and Fox (1995) and Morisio et al. (2002) showed that the construction and use of repositories is important, but not fundamental to the success of reuse programs. In this research, we analyzed the use of component repositories in the software organizations, aiming to analyze their impact on reuse programs.

Analyzing the survey, we identified that the existence of component repositories does not assure the success in reuse

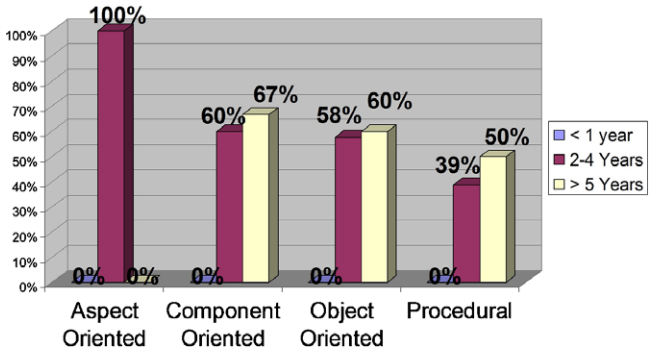

Fig. 9. Reuse success × Development approach × Team experience.

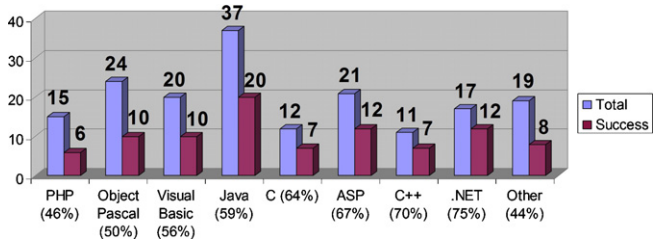

Fig. 10. Influence of the programming language on reuse success.

levels. In fact, we observed the opposite effect: the percentage of success in organizations without a repository was superior to the ones that have one.

We verified that the organizations that had repositories considered mainly: (i) source code; (ii) software components; (iii) document models; (iv) reports and screen generators; and (v) data base queries.

The reasons for the success in organizations with component repositories were not identified, so a subsequent study is necessary. However, it was identified that some software organizations store their assets in simple structures like Concurrent Versions System (CVS), without an efficient search mechanism. It is possible that this may have contributed to failure in the efforts of promoting reuse.

Based on these observations, we conclude that the existence of a reuse repository has **no influence** on software reuse success.

3.4.4. CASE tools usage

There is a vast literature related to CASE tools and software reuse. Also, their usage is growing inside software development organizations. In this survey, we tried to verify if the common belief held by many organizations – that CASE tools help to increase reusability – is true for the Brazilian scenario.

Frakes and Fox (1995) stated that CASE tools are not effective in promoting reuse. One possible explanation for this problem is the incorrect usage of these tools. In our study, we attempted to verify if the Brazilian organizations are using CASE tools in their processes, and if this is contributing to success in reuse.

The collected data showed that 71% (17 out of 24) of the organizations that claimed to use CASE tools in their projects reported reuse success, against 54% (7 out of 13) of success for the organizations that do not use CASE. Therefore, we conclude that CASE has a **strong** influence on reuse success. However, it must be stressed that we did not analyze what kind of tools were used, and which ones were the responsibles for the success in reuse. But the fact that CASE is present on the organizations where reuse was successful indicates that they should be considered if a reuse program is to be adopted.

3.5. Processes factors

3.5.1. Quality models usage

Quality and maturity models are used to define and evaluate the process that is used by an organization, in order to assure the quality of the final product (Chrissis et al., 2003). As expected, quality model usage is a factor that contributed with the software reuse success: 71% (12 out of 17) of the software organizations that adopt a quality model obtained success in software reuse, while only 45% (18 out of 40) of the software organizations that did not adopt a quality model managed to achieve success in reuse.

Therefore, we conclude that quality models usage has a **strong** influence on reuse success.

On the other hand, the kind of model adopted did not have any importance, with little difference between the software organizations that adopted ISO 9001/9000-3 or CMMI 2. None of the software organizations that were surveyed had CMMI 3 or higher (two software organizations were in preparation to obtain level 3, though) and the software organizations that adopted other quality models did not specify which ones they adopted.

3.5.2. Systematic reuse process

The analysis of the quality model usage (Section 3.5.1) has already shown that software organizations that adopt some model obtain higher success in software reuse. Besides, Frakes and Fox (1995) and Morisio et al. (2002) concluded that the usage of a specific reuse process, or the adaptation of some software development process into the context of reuse, are aspects that have a strong influence in the success of reuse.

As shown by our survey, this conclusion is also valid for the Brazilian scenario. The results have shown that 73% (11 out of 15) of the organizations that adopted a systematic reuse process claimed to achieve success in reuse, against 45% (19 out of 42) of success in software organizations that did not adopt any reuse process. This confirms the results presented in Frakes and Fox (1995), Morisio et al. (2002), Rine (1997). Therefore, we conclude that the systematic reuse process adoption is a superior factor, with a **strong** influence on software reuse success.

3.5.3. Kind of reused assets

Several assets can be reused, such as: requirements specification, architecture, application design, tests and source code (Krueger, 1992). However, many organizations believe that the source code is the only asset that can be reused in different contexts. However, contradicting this idea, our study has shown that software organizations claimed to reuse other kinds of assets as well. Fig. 11 summarizes the answers regarding this question.

Among the software organizations that participated of the survey, only two did not reuse the source code, and one obtained success with reuse. 9% (5 out of 52) of the organizations reuse all the assets presented early and all obtained success; 12% (7 out of 52) of the organizations reuse source code and tests specifications; all of them obtain success too. At the end, between the software orga-

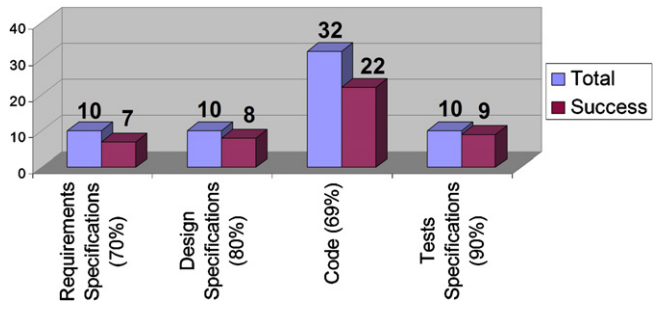

Fig. 11. Influence of the kind of reused assets on reuse success.

nizations that reuse just the source code – 17% (9 out of 52) of the total – only 56% (5 out of 9) had success.

In this way, we can conclude that the kind of reused assets has a **strong** influence on reuse success. More specifically, software reuse should be considered during all the life-cycle development and not just in the implementation step.

3.5.4. Origin of the reused assets

There are some different opinions about what should be the origin of the reusable assets. Initially, the general belief is that the reusable assets must be built from scratch, aiming the maximum of abstraction so they can be used in various scenarios. However, this is an expensive process, usually not adopted in the industry due to budget and schedule limitations. Besides, organizations that do not work with specific domains have more difficulties in designing assets that can be used in more than one project (see Section 3.3.1). Thus, some assets are built from existent products, through an exhaustive reengineering process, or simply by adapting existing products with a small effort.

Analyzing the data about the origin of the reusable assets, we observed that 62% (8 out of 13) of the organizations reusing assets built from scratch obtained success in reuse, against 71% (22 out of 31) of success in the organizations reusing assets extracted from existing products. Therefore, since the success rate is only slightly different (9%) we conclude that the origin of the reuse assets has a **weak** influence on software reuse. More specifically, assets extracted from some existing product have greater chance of leading to reuse success. In this sense, the thought that developing reusable assets from scratch is the ideal solution turned out not to be true in our study.

These results are somehow related to the observations made in Section 3.3.1. When developing software based on a product family, the reusable assets are more closely related to a specific product. Since the product family approach turned out to be more successful, it makes sense that reusing assets from some existing product will be more successful than reusing assets built from scratch, with less focus on some product. Also, assets obtained from pre-existing products, in spite of demanding an effort to become reusable, have already been built, tested and validated, assuring their quality and justifying their success.

3.5.5. Previous development of reusable assets

Usually, the reusable assets can be developed in two moments in a project: before the development or in parallel with the development. Building the assets before the development usually produces a better design and better internal structure. However, due to time constraints this can not be performed every time, and these assets end up being developed in parallel with the system (Frakes and Isoda, 1994).

The study showed that the development of reusable assets before the application development is a good practice. 85% (17 out of 20) of the software organizations that followed this approach obtained success in reuse, while only 43% (6 out of 14) of the software organizations that developed the assets in parallel with the system achieved success. We conclude that the previous development of reusable assets has a **strong** influence on reuse success.

3.5.6. Specific function in the software reuse process

Sections 3.5.1 and 3.5.2 presented that the adoption of a quality model and a systematic reuse process strongly contribute to software reuse success. In our survey, we tried to verify if the creation of specific functions associated to these processes can contribute to the success of reuse programs.

The analyzed data showed that only a few organizations had specific functions associated to the reuse process: only 12% (7 out of 57) of the total. Besides, it was verified that the number of organizations that obtained success with reuse is bigger in the organizations that do not have associate functions with reuse process than the ones that have it. 73% (19 out of 26) of the organizations that do not have a reuse-specific function reported success in reuse, against 57% (4 out of 7) of success reported by the organizations with a reuse-specific function in their processes.

We found these results unexpected, because theoretically the existence of specific functions should facilitate reuse. Additionally, the number of participant organizations with specific functions associated to the reuse process is too small (7), and therefore we think that further research is needed in order to verify why the expected results are not being achieved. Therefore, we found **no data** associating the existence of reuse-specific functions and reuse success.

3.5.7. Software reuse measurement

The measurement of the reuse levels is important to the real definition of the success in reuse programs, as well as to the identification of faults or improvements (Poulin, 1997; Mascena et al., 2005; Aggarwal et al., 2005; Poulin, 2006).

In spite of this, only one organization claimed to have a program for measuring the reuse level, called *Return Over Assets*, and it did not have success with reuse. Other four organizations were in the process of planning the introduction of measurement programs; however, it was not possible to identify which program these will be.

Thus, we conclude that there were **no data** to identify if software reuse measurements are predictive of reuse success.

This also had influence in our study. Since the surveyed organizations had no reuse measurements, our conclusions were based on a subjective analysis of the answers provided by the participants, as explained in Section 2.4.1.

3.5.8. Software certification process

One of the most important issues in software reuse is to assure the quality of the assets (Alvaro et al., 2005) in order to obtain the desired benefits, such as defect reduction. When reusing an asset, one needs to be convicted that the asset has a determinate quality level, which can be done

through certification. In our study, we tried to identify if the usage of a well-defined certification process or mechanism has correlation with reuse success.

It was possible to identify that asset certification processes are not widely used in Brazil, with only 12% (7 out of 57) of the total of surveyed organizations claiming to have one. As asset certification methods, the organizations highlighted: (i) certification process introduced in the development methodology; (ii) assets being used in pilot applications; (iii) adoption of systematic final tests; and (iv) approval by the product committee. among these organizations, the ones that adopted (iii) and (iv) had more success.

The data analysis showed that the success level among the organizations that adopted this kind of process (71%, 5 out of 7) is practically the same as in the software organizations that did not adopt it (69%, 18 out of 26). However, this number is too small to extract any meaningful conclusion. Moreover, certification is a new area, so even for the organizations that have a certification process, we do not know how useful it is and how this process is carried out. Thus, we conclude that this study found **no data** for evaluating if a certification process has some influence on reuse success, and further research is necessary in order to better establish such relationship.

3.5.9. Configuration management of the reusable assets

Configuration management processes are fundamental to help in maintaining the assets quality during their evolution. In the reuse context, it is even more important, since modifications in some asset can require modifications not only in one project, but in all projects that use it. Thus, the study tried to identify how Brazilian organizations are using the configuration management processes for reusable assets, as well as the impact on the reuse success.

Analyzing the survey, it was visible that the configuration management process for reusable assets is not widely used: only 14% (10 out of 57) of the organizations use it, and most of them had a version control tool, such as CVS, as the main management model.

We also observed that 80% (8 out of 10) of the software organizations that adopted it obtained success in reuse programs, while only 65% (15 out of 23) of organizations that do not adopt it obtained success in reuse. Thus, we conclude that configuration management is an important issue, with **strong** influence on reuse success.

3.6. Summary of the study

Table 1 summarizes the influence of the studied factors on software reuse success.

As it can be seen, the most influencing factors are: *Independent reusable assets development team*, *Product family approach*, *CASE tools usage*, *Quality models usage*, *Systematic reuse process*, *Kind of reused assets*, *Previous development of reusable assets* and *Configuration management of the reusable assets*.

Table 1
Factors that can influence reuse programs

Factor	Influence			
	Strong	Weak	None	No data
Software organizations and team size			X	
Project team experience		X		
Software reuse education			X	
Rewards and incentives				X
Independent reusable assets development team	X			
Product family approach	X			
Kind of software developed			X	
Application domain		X		
Software development approach		X		
Programming language		X		
Repository systems usage			X	
CASE tools usage	X			
Quality models usage	X			
Systematic reuse process	X			
Kind of reused assets	X			
Origin of the reused assets		X		
Previous development of reusable assets	X			
Specific function in the software reuse process				X
Software reuse measurement				X
Software certification process				X
Configuration management of the reusable assets	X			

Also influencing the reuse success, but in a smaller degree, are the following factors: *Application domain*, *Software development approach*, *Programming language* and *Origin of the reused assets*.

For the remaining factors, we found no data that relates the factor with reuse success. This indicates that this studied failed to establish a relationship between the factor and software reuse, and further research is needed, or that the factor has no influence on software reuse.

4. Related works

As explained in the beginning of the paper, the results of this survey are not restricted to Brazilian software organizations. In order to support this, we identified some similar works involving organizations in US and Europe, and compared their results with ours. After an extensive review on the software reuse field, three similar studies were found.

Frakes and Fox (1995) present an empirical study performed in 28 software organizations from US and 1 from Europe, with 16 questions about software reuse. This study was based in the premise that software organizations are implementing systematic reuse programs and need some practical answers. The questions involved important issues, such as repository systems, education, legal problems and CASE tools, among others.

In their study, the factors influencing reuse are: type of industry, perceived economic feasibility, high quality

assets, common software process, and reuse education. On the other hand, Frakes & Fox suggest that the adopted programming language, the CASE tool usage, the team experience, the rewards policy adoption, the legal problems, the repository introduction, the organization size, quality issues, the adopted reuse metrics and the assumption that the developers prefer to implement instead of reuse, do not directly affect the reuse level in the organizations.

Rine et al. (1998) presents the results of a survey involving 109 software organizations from five different countries. Their research supported the thesis that there is a set of success factors which are common across organizations and have some predictability relationships to software reuse.

According to Rine & Sonnemann, organizations with the highest reuse capability have the following: product-line approach, architecture which standardizes interfaces and data formats, common software architecture across the product-line, design for manufacturing approach, domain engineering, with a separate team for software reuse, reuse process, management which understands reuse issues, software reuse advocate(s) in senior management, state-of-the-art tools and methods, precedence of reusing high level software artifacts such as requirements and design versus just code reuse, and trace end-user requirements to the components (systems, subsystems, and/or software modules) which support them.

They also state that the following factors have a weak relationship with reuse: the use of the best software engineers to develop (team experience), use of a pilot project for determining software reuse best practices, and having projects documents about the software reuse lessons learned.

On the other hand, according to the authors, the following factors have no influence on software reuse: Use of a library (repository approach), performing market analysis for components, establishing candidates library to eliminate duplication in projects, who develops the reusable components (application engineering or domain engineering), reuse education and component certification.

Morisio et al. (2002) describe an empirical study based on 24 projects that were developed in European software organizations between 1994 and 1997. Its main goal was to identify success and failure factors for reuse programs. According to the authors, the most important factors for a reuse program are: *Systematic reuse processes*, *top management commitment*, *human factors and the product family approach*.

For Morisio et al., the following factors have no influence on reuse success: specific reuse roles, repository usage, organization and team size, process maturity (quality model), type of software being developed, and the development approach.

Additionally, they report three main causes of reuse failure: not introducing reuse-specific processes, not modifying nonreuse processes, and not considering human factors. The root cause was the lack of top management commitment, or non awareness of the importance of those factors, often coupled with the belief that using the object-oriented approach or setting up a repository is all that is needed to achieve success in reuse.

Table 2
Comparison with related works: Yes = some influence on reuse; No = no influence on reuse; N/D = no data

Factor	Influences reuse?			
	This	Frakes	Rine	Morisio
Software organizations and team size	No	No	N/D	No
Project team experience	Yes	No	Yes	N/D
Software reuse education	No	Yes	No	N/D
Rewards and incentives	N/D	No	N/D	N/D
Independent reusable assets development team	Yes	N/D	Yes	N/D
Product family approach	Yes	N/D	Yes	Yes
Kind of software developed	No	Yes	N/D	No
Application domain	Yes	Yes	N/D	N/D
Software development approach	Yes	N/D	N/D	No
Programming language	Yes	No	N/D	N/D
Repository systems usage	No	No	No	No
CASE tools usage	Yes	No	Yes	N/D
Quality models usage	Yes	Yes	N/D	No
Systematic reuse process	Yes	Yes	N/D	Yes
Kind of reused assets	Yes	N/D	Yes	N/D
Origin of the reused assets	Yes	N/D	N/D	N/D
Previous development of reusable assets	Yes	N/D	N/D	N/D
Specific function in the software reuse process	N/D	N/D	N/D	No
Software reuse measurement	N/D	No	N/D	N/D
Software certification process	N/D	No	No	N/D
Configuration management of the reusable assets	Yes	N/D	N/D	N/D

These studies do not examine the exact same factors as ours. But the factors examined in common share a high level of similarity, as shown in Table 2.[6]

Except for two factors (Software development approach and Programming language), our results have led to the same conclusions as at least one related work. Some other factors share the exact same conclusions, mainly:

- *Product family approach*: Most authors agree that the product family approach can lead to the desired reuse benefits of quality and productivity;
- *Systematic reuse process*: The need for a systematic reuse process is also highlighted by these surveys, as already stated by the most important authors on the software reuse area; and
- *Repository systems usage*: All studies agree that the use of a repository system have no influence on reuse success.

[6] Only the first author of each study is shown, in order to increase the table's readability.

Another important factor present in all other studies is the organization management involvement. In our study, this factor was the main motivation, since the guide that is presented in the next section is intended for managers interested in obtaining a path toward successful reuse. So, we assume that management involvement is mandatory for organizations seeking reuse.

5. Key factors for introducing software reuse in software organizations

In the previous sections, we presented the factors that influence the success of reuse programs in Brazilian organizations, comparing them to the reality of other countries. In this section, we present the key points for introducing or improving reuse processes in a software organization, according to its size (small, medium or large). *It must be stressed that the key points are based on the survey results. In this sense, they are not conclusive and cannot be generalized. However, they present good warnings for organizations to review their processes or to adopt new strategies for software reuse*.

In order to develop this guide, we calculated the percentage of success for each factor again, but now considering only small, medium and large organizations, separately. Then we repeated the analysis for each organization size, observing the variation of reuse success in each factor, as well as other qualitative data, including their comments on the questionnaire. We then prepared a guide for each kind of organization, by presenting which factors should be considered in each case, according to the four categories considered: organizational factors, business factors, technological factors and processes factors.

5.1. Small software organizations

In our study, a organization is considered small if it has less than 50 employees. These organizations had a good reuse success rate, of 64% (16 out of 25).

Organizational factors: in small organizations, the only organizational factor that seems to have a little influence on reuse success is the development team experience. 75% (6 out of 8) of the organizations with professionals with more than 5 years of experience had success, against 63% (10 out of 16) of organizations with professionals with less than 5 years of experience. The other factors did not presented large variations of reuse success percentage, indicating that they have no influence on reuse success.

Business factors: according to our results, the business factors have little influence on reuse success in small organizations. The product family approach, which according to the overall results have a strong influence on reuse success, presents no advantage for small organizations. 67% (6 out of 9) of the organizations developing isolated products claimed reuse success, against the exact same 67% (10 out of 15) of the organizations focusing on product families. The only factor with some influence was the application domain, with reuse percentages varying from 57% (4 out of 7, in the educational domain) to 100% (7 out of 7, in telecommunications and development tools domains).

Technological factors: the object-oriented approach had a little advantage in terms of reuse success in small organizations, with 79% (15 out of 19) of success in organization adopting it. The other development approaches resulted in smaller percentages of success, such as 50% (7 out of 14) for the procedural approach. CASE support was also present in most successful small organizations, with 89% (8 out of 9) of success. The other factors had no influence in reuse success.

Processes factors: the adoption of a quality model and the existence of a systematic reuse process had strong influence on small organizations, with 86% (6 out of 7) of claimed reuse success, against 59% (10 out of 17) of success in organizations without any quality model and without a systematic reuse process. The previous development of reusable artifacts also presented a high level of success in small organizations, with 91% (10 out of 11) of success, against 57% (4 out of 7) of success for organizations that do not develop the reusable assets previously. Finally, configuration management of the reusable assets accounted for 100% (5 out of 5) of reuse success, against 69% (9 out of 13) for organizations that do not perform it.

Summary: *In order to have good results with software reuse, small organizations should have experienced professionals in their teams. Small organizations should also be aware that the application domain may influence reuse success. Object-oriented development and CASE support should also be considered. The focus on the process is also important, mainly the existence of a quality model and a systematic reuse process. Whenever possible, reusable assets should be previously developed, and a configuration management process should guarantee proper evolution of these assets.*

5.2. Medium software organizations

In our study, a software organization is considered medium if it has between 50 and 200 employees. These software organizations had the lowest level of reuse success, only 27% (3 out of 11).

Organizational factors: in medium software organizations, the team experience had some influence on reuse success: 40% (2 out of 5) of reuse success in organizations with experienced (more than 5 years of experience) teams, against 14% (1 out of 7) for teams less experienced. The existence of an independent development team for reusable assets also seems to have some influence on reuse success, although the amount of data is too small in this case to obtain any conclusive conclusion: 100% (3 out of 3) of success in this situation, against no success (0 out of 3) in organizations without an independent reuse team.

Business factors: the product family approach is more successful in medium organizations, with 50% (2 out of 4) of success, against 13% (1 out of 8) in organizations that

work with isolated products. The other business factors had no influence on medium organizations, according to our results.

Technological factors: the only technological factor that seems to be related to reuse success in medium organizations is the use of CASE tools: 50% (2 out of 4) of success in medium organizations using some CASE tool, against no success (0 out of 1) in organizations without CASE support.

Processes factors: in medium organizations, the existence of a systematic reuse process should be considered. Our results show this fact not only through the higher percentage of reuse success in organizations that have a systematic reuse process (50%, 1 out of 2), but due to the low percentage of success in organizations that do not have one (20%, 2 out of 10). The kind of reused assets is also important, with organizations reusing all kinds of assets having more success (80%, 4 out of 5) than organizations reusing only source code (33%, 1 out of 3). Reusing assets obtained from existing products was also more important than reusing assets developed from scratch, with 50% of success (3 out of 6) against no success at all(0 out of 1), respectively. The previous development of reusable artifacts also presented a high level of success in medium organizations, with 67% (2 out of 3) of success, against no success (0 out of 2) in organizations that do not develop the reusable assets previously. Finally, medium organizations with a configuration management process for the reusable assets had 100% (1 out of 1) of success, against 25% (1 out of 4) of success in organizations without it.

Summary: *Aiming to obtain good results with software reuse, medium organizations should have experienced teams and, if possible, a separate team dedicated to reuse activities. They should focus on the development of product families, through a systematic reuse process. All kinds of assets should be reused, and not just source code, and these should be obtained from existing products, instead of developed from scratch. Whenever possible, the reusable assets should be created before the development of the products, aiming to create useful reusable abstractions. CASE support is also advised, together with a good configuration management support.*

5.3. Large software organizations

The organization is considered large if it has more than 200 employees. This group had a good level of reuse success, 52% (11 out of 21).

Organizational factors: according to our survey, organizational factors do not have great influence on reuse success in large organizations, except for the existence of an independent team for developing reusable assets, which obtained 100% (3 out of 3) of success, against 56% (5 out of 9) of success in organizations without such team.

Business factors: the product family approach was adopted by the most successful organizations. 63% (5 out of 8) of the organizations that adopted a product family approach had success, against 42% (5 out of 12) of success for the organizations that develop isolated products. We observed no influence from the other factors, except for the kind of software developed. The organizations that develop real-time systems had no reuse success, indicating that it is very difficult to reuse assets from this kind of software.

Technological factors: 100% (3 out of 3) of the large organizations that use component-based development had success in reuse. Object-oriented approach was also successful, with 59% of success (10 out of 17). On the other hand, the procedural approach was less successful, with 44% (7 out of 16) of success. The programming language also had some influence, with the most successful ones being C++ (75%, 3 out of 4),.NET (83%, 5 out of 6) and ASP (83%, 5 out of 6).

Processes factors: the use of a quality model is an important factor. Organizations with some quality model had 86% (6 out of 7) of success, against 36% (5 out of 14) of success in organizations that do not have one. The existence of a systematic reuse process was also important, with 67% (4 out of 6) of success against 47% (7 out of 15). When combined, the existence of a quality model and a systematic reuse process resulted in 100% (21 out of 21) of success. The reuse of all kinds of assets had also some influence, with 77% (10 out of 13) of success against 64% (7 out of 11) of success in organizations that reuse only source code. Moreover, the reusable assets should be constructed before the product development, with 83% (5 out of 6) of success in these cases, against 40% (2 out of 5) of success in the cases where the reusable assets are developed together with the product.

Summary: *In order to have good results with software reuse, large organizations should have an independent team for the development of reusable assets. It should have a strong focus on the development of product families, through a systematic reuse process. The adoption of a quality model is also advised. It should promote reuse of all kinds of assets, and not just source code. The reusable assets should be*

Table 3
Key factors for reuse programs in small, medium and large organizations

Factor	Organization size		
	Small	Medium	Large
Project team experience	X	X	
Independent reusable assets development team		X	X
Product family approach		X	X
Application domain	X		
Software development approach	X		X
Programming language			X
CASE tools usage	X	X	
Quality models usage	X		X
Systematic reuse process	X	X	X
Kind of reused assets		X	X
Origin of the reused assets		X	
Previous development of reusable assets	X	X	X
Configuration management of the reusable assets	X	X	

The factors that have no influence were not included in the table.

previously developed, if possible, using the component-based and object-oriented approaches. The programming languages should also be considered, with priority to object-oriented languages such as C++ and .NET.

5.4. Summary of the key factors for introducing software reuse in software organizations

Table 3 summarizes the key factors described in the previous sections.

6. Concluding remarks and future directions

Software reuse is a key aspect for software organizations interested in improving time-to-market, costs and quality. Since Mcllroy's work (McIlroy, 1968) until the current days (Moore, 2001), software organizations are seeking practical ways to adopt software reuse. However, before becoming ready to obtain the desired benefits, they need to introduce a reuse program in their agenda, and it should have a low risk.

In this paper, we presented the results of a survey involving Brazilian organizations, that helped in identifying potential problems and indicate viable solutions. Although only Brazilian organizations were surveyed, we showed that the results are valid for other countries as well, through a comparison with important related works.

The paper also shows an analysis of the key factors of software reuse, providing a useful guide that managers can use when introducing reuse programs in small, medium and large software organizations. Researchers may also use the results of this survey to better investigate the factors that were left without conclusions due to the lack of information.

One particularly important factor is the product family approach, which appeared as having strong influence on software reuse. However, it was beyond the scope of this survey to further investigate which product family activities are more important, or if the mere notion of developing with product families in mind is sufficient to achieve high reuse levels. Future works may investigate which practice areas (Clements and Northrop, 2002) should be more carefully taken into consideration when adopting such approach.

As another future work, a new project is to be started with the Brazilian Government support, with 2 years, with the goal of studying Brazilian organizations in general, and not just concerned with software reuse. We believe that our study will greatly contribute in this direction.

Acknowledgement

The authors thank the organizations that answered the survey.

References

Aggarwal, K.K., Singh, Y., Kaur, A., Malhotra, R., 2005. Software reuse metrics for object-oriented systems. In: Third ACIS International Conference on Software Engineering Research, Management and Applications. IEEE/CS Press, MI, USA, pp. 48–54.

Almeida, E.S.d., Alvaro, A., Lucrédio, D., Garcia, V.C., Meira, S.R.d.L., 2004. RiSE project: towards a robust framework for software reuse. In: IEEE International Conference on Information Reuse and Integration (IRI). IEEE/CMS, Las Vegas, USA, pp. 48–53.

Almeida, E.S.d., Alvaro, A., Lucrédio, D., Garcia, V.C., Meira, S.R.d.L., 2005. A survey on software reuse processes. In: IEEE International Conference on Information Reuse and Integration (IRI 2005). IEEE/CS Press, Las Vegas, USA.

Alvaro, A., Almeida, E.S.d., Meira, S.R.d.L., 2005. Software component certification: a survey. In: IEEE Euromicro Conference – CBSE Track. SAGE, Berlin.

Anastasopoulos, M., Gacek, C., 2001. Implementing product line variabilities. In: Symposium on Software Reusability (SSR), Toronto, Canada, pp. 109–117.

Bayer, J., Flege, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., Widen, T., DeBaud, J., 1999. PuLSE: a methodology to develop software product lines. In: Symposium on Software Reusability (SSR). ACM Press, Los Angeles, USA, pp. 122–131.

Burégio, V.A.d.A., Almeida, E.S.d., Lucrédio, D., Meira, S.R.d.L., 2007. Specification, design and implementation of a reuse repository. In: 31st IEEE Annual International Computer Software and Applications (COMPSAC) Conference – Short paper, Beijing, China.

Chrissis, M.B., Konrad, M., Shrum, S., 2003. CMMI: Guidelines for Process Integration and Product Improvement. Addison-Wesley.

Clements, P., Northrop, L., 2002. Software Product Lines: Practices and Patterns. Addison-Wesley.

Deursen, A.v., Klint, P., Visser, J., 2000. Domain-specific languages: an annotated bibliography. ACM SIGPLAN Notices 35 (6), 26–36.

Endres, A., 1993. Lessons learned in an industrial software lab. IEEE Software 10 (05), 58–61.

Fichman, R.G., Kemerer, C.F., 2001. Incentive compatibility and systematic software reuse. The Journal of Systems and Software 57 (1), 45–60.

Frakes, W.B., Fox, C., 1995. Sixteen Questions about Software Reuse. Communications of the ACM 38 (06), 75–87.

Frakes, W.B., Isoda, S., 1994. Success factors of systematic software reuse. IEEE Software 11 (01), 14–19.

Frakes, W.B., Kang, K., 2005. Software reuse research: status and future. IEEE Transactions on Software Engineering 31 (7), 529–536.

Griss, M., 1995. Making software reuse work at Hewlett-Packard. IEEE Software 12 (01), 105–107.

Guo, J., Luqi, 2000. A survey of software reuse repositories. In: 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems. IEEE/CS Press, Edinburgh, Scotland, p. 92.

John, I., Knodel, J., Lehner, T., Muthig, D., 2006. A practical guide to product line scoping. In: 10th International Software Product Line Conference (SPLC'06). IEEE/CS Press, Baltimore, USA, pp. 3–12.

Joos, R., 1994. Software reuse at Motorola. IEEE Software 11 (05), 42–47.

Kitchenham, B., Pleeger, S., 2002. Principles of survey research. Part 2: Designing a survey. ACM SIGSOFT Software Engineering Notes 27 (1), 18–20.

Krueger, C., 1992. Software reuse. ACM Computing Surveys 24 (02), 131–183.

Krueger, C., 2006. New methods in software product line development. In: 10th International Software Product Line Conference (SPLC'06). IEEE/CS Press, Baltimore, USA, pp. 95–102.

Lucrédio, D., Almeida, E.S.d., Prado, A.F.d., 2004. A survey on software components search and retrieval. In: Steinmetz, R., Mauthe, A. (Eds.), 30th IEEE EUROMICRO Conference, Component-Based Software Engineering Track. IEEE/CS Press, Rennes – France, pp. 152–159.

Lynex, A., Layzell, P.J., 1998. Organisational considerations for software reuse. Annals of Software Engineering 5, 105–124.
Mascena, J.C.C.P., Almeida, E.S.d., Meira, S.R.d.L., 2005. A comparative study on software reuse metrics and economic models from a traceability perspective. In: IEEE International Conference on Information Reuse and Integration (IRI). IEEE/CS Press, Las Vegas, NV, USA.
McIlroy, M.D., 1968. Mass produced software components. In: NATO Software Engineering Conference, pp. 138–155.
Mili, R., Mili, A., Mittermeir, R.T., 1997. Storing and retrieving software components: a refinement based system. IEEE Transactions on Software Engineering 23 (7), 445–460.
Mili, A., Mili, R., Mittermeir, R.T., 1998. A survey of software reuse libraries. Annals of Software Engineering Notes 5, 349–414.
Moore, M., 2001. Software reuse: silver bullet? IEEE Software 18 (5), 86.
Morisio, M., Ezran, M., Tully, C., 2002. Success and failure factors in software reuse. IEEE Transactions on Software Engineering 28 (04), 340–357.
Paiva, D.M.B., Fortes, R.P.d.M., 2005. Design rationale in software engineering: a case study. In: Seventeenth International Conference on Software Engineering and Knowledge Engineering, Taipei, Taiwan, Republic of China, pp. 342–348.
Pohl, K., Bockle, G., Linden, F.V.D., 2005. Software Product Line Engineering. Springer-Verlag.
Poulin, J., 1997. Measuring Software Reuse. Addison-Wesley.
Poulin, J., 2006. The business case for software reuse: reuse metrics, economic models, organizational issues, and case studies. In: 9th International Conference on Software Reuse – Tutorial Notes, Torino, Italy.
Rine, D., 1997. Supporting reuse with object technology. IEEE Computer 30 (10), 43–45.
Rine, D., 1997. Success factors for software reuse that are applicable across domains and businesses. In: ACM Symposium on Applied Computing. ACM Press, San Jose, CA, USA, pp. 182–186.
Rine, D., Sonnemann, R., 1998. Investment in reusable software. A study on software reuse investment success factors. The Journal of Systems and Software 41, 17–32.
Rothenberger, M.A., Dooley, K., Kulkarni, U., Nada, N., 2003. Strategies for software reuse: a principal component analysis of reuse practices. IEEE Transactions on Software Engineering 29 (09), 825–837.
Seacord, R.C., 1999. Software engineering component repositories. In: International Workshop on Component-Based Software Engineering, Held in conjunction with the 21st International Conference on Software Engineering (ICSE), Los Angeles, CA, USA.
Sherif, K., Appan, R., Lin, Z., 2006. Resources and incentives for the adoption of systematic software reuse. International Journal of Information Management 26 (1), 70–80.

Daniel Lucrédio graduated and got his M.Sc. degree in Computer Science at the Federal University of São Carlos, and is currently a Ph.D. candidate at the University of São Paulo – São Carlos, Brazil, with a period as a visiting scholar at George Mason University, Virginia, USA. He is the author of MVCASE, a free tool for UML modeling and component-based development, three times awarded at the Brazilian Symposium on Software Engineering, in 2001, 2002 and 2004. Author of several papers in important vehicles, such as IEEE and Euromicro conferences, in the software reuse and component-development areas, he presented research works and speeches at conferences in Brazil and several other countries, including Canada, United States, France and Hong Kong. He also works as a software development coordinator and Java programmer since 1997.

Kellyton dos Santos Brito is member of Brazilian Computer Society, member of the RiSE – Reuse in Software Engineering – Group, and a Researcher of C.E.S.A.R. - Recife Center of Advanced Studies and Systems. He received his B.Sc. in computer science from the Federal University of Bahia in 2005 and his M.Sc. from the Federal University of Pernambuco in 2007. He works with research and development of innovative solutions, and also participated in projects for mobile devices development (phones and PDAs), using PDAs and wireless in medical institutions and digital images for medical diagnosis, among others. Currently, Kellyton performs research in the software engineering and reuse areas, focusing on reverse engineering and knowledge extraction from legacy code.

Alexandre Alvaro is a senior member of the RiSE (Reuse in Software Engineering) group, at the Federal University of Pernambuco. He received his M.Sc. in computer science and he is a Ph.D. candidate in Computer Science at the Federal University of Pernambuco. Nowadays, he is a Systems Engineer and Software Reuse Consultant at the Recife Center for Advanced Studies and Systems (C.E.S.A.R.). Author of several papers in important vehicles, including topics such as software reuse, component-based development and software component certification. He is involved in four industrial projects that focus on different aspects of reuse, such as processes, environments and tools.

Vinicius Cardoso Garcia is a member of the Brazilian Computer Society, and a senior member of the RiSE - Reuse in Software Engineering – Group, at the Federal University of Pernambuco. He received his B.Sc. in computer science from the Salvador University (UNIFACS) in 2001, and the M.Sc. in computer science from Federal University of São Carlos in 2005. He is currently a Ph.D. candidate at the Federal University of Pernambuco since 2005, and also a Systems Engineer at the Recife Center for Advanced Studies and Systems (C.E.S.A.R.) since 2005. Vinicius Garcia is (co-)author of over 30 referenced publications presented at conferences such as WCRE, IRI, ECOOP, CBSE, ICSR and EUROMICRO, amongst others. He is currently involved in four research projects in the computer Science area, more specifically in Software Reuse Maturity Models and Software Reuse Adoption in Software Development Process.

Eduardo Santana de Almeida is a reuse consultant at Recife Center for Advanced Studies and Systems (C.E.S.A.R), where he has conducted more than five industrial projects focused on the various reuse aspects, such as methods, processes and tools. Nowadays, he is a member of Consulters Program, a group composed of seventeen people out of the 650 employees of C.E.S.A.R, where his responsibility is the software reuse area. In the reuse efforts, Dr. Almeida has worked as researcher involved since the conception solutions until their deployment. He is also the author of about 70 papers published and presented in conferences on these and other software reuse topics in leading venues worldwide and has served as referee in important journal papers with focus on reuse such as IEEE Transactions on Software Engineering and Journal of Systems and Software. He is also head of the Reuse in Software Engineering (RiSE) where one of his activities is to start new cooperation projects around the word involving software reuse and Member of the Brazilian Committee composed of 50 specialists in the software component area, created by Brazilian Government. Dr. Almeida was Co-Chair of the 6th Brazilian Workshop on Component-Based Development (2006), creator of the WIRE – Workshop for Reuse Introduction in Companies where he has served as Co-Chair and Member of the Steering Committee, creator of the RISS – RiSE Summer School on Software Reuse where he has served as Director and Guest Editor of the Journal of the Brazilian Computer Society, Special Issue on Software Reuse (2008). At the Porto Digital, Dr. Almeida, in conjunction with the RiSE group, has been the main vehicle to introduce reuse ideas in industrial and academic environment, being responsible also to define the M.Sc. and Ph.D. courses focused on software reuse at the Federal University of Pernambuco and C.E.S.A.R. In the last years, Dr. Almeida has been Visiting Researcher in important groups working with software reuse such as University of Mannheim (Colin Atkinson's Group) and Eidgenössische Technische Hochschule (ETH) (Bertrand Meyer's Group). He has gained two awards of the Brazilian Symposium on Software Engineering - Tools Session (2002, 2004) and is also Member of the IEEE Computer Society and the Brazilian Computer Society. In addition he was leader of the conception on the first open source book on software reuse around the world and has worked as independent consultant involved in training, mentoring, consultant and solution development focused on

software reuse. Dr. Almeida is also IEEE Computer Society Certified Software Development Professional (CSDP).

Renata Pontin de Mattos Fortes is an Associate Professor with the Universidade de São Paulo (USP), Brazil. She holds a B.Sc. in Computer Science from USP, a M.Sc. in Computer Science from USP and a Ph.D. degree in Computational Physics at IFSC-USP, under Prof. Dr. Alvaro Garcia Neto. Renata Fortes has concluded her post-doctoral visit at Georgia Tech supervised by Prof. Dr. Gregory Abowd. Fortes's research investigates problems associated with open source process and software reuse – in most cases, looking for how design rationale can be gather and be available during the engineering process. Dr. Renata Fortes has supervised 15 M.Sc., 1 Ph.D. and 26 undergraduate students. Fortes coordinated the SAFE Project, in collaboration with Universidade de Mato Grosso do Sul and Async Open Source funding from FINEP. She has collaborated in other relevant projects, funding from CNPq and FAPESP in Brazil and NSF in the US. She has recently been head of the WebMedia Brazilian Conference hosted in Minas Gerais (2005). She has also participated in the program committee for other Conferences (SBES, SIGDOC, SIGCSE). Author of papers in important vehicles, such as IEEE and ACM conferences, in the software reuse and knowledge-engineering areas, she presented works and speeches at conferences in Brazil and other countries, including Taiwan, Scotland, US, France and Portugal. Her research interests include Software Engineering, Human-Computer Interaction and Hypermedia.

Silvio Lemos Meira is the chair of Software Engineering at the Center for Informatics at the Federal University of Pernambuco in Recife, Brazil. Prof. Meira holds a B.Sc. in Electronic Engineering from ITA, S. J. Campos, Brazil, an M.Sc. in Computer Science from the Federal University of Pernambuco and a Ph.D. in Computing, under David Turner, from the University of Kent at Canterbury, UK. Among many other responsibilities, Prof. Meira is the leader of the Reuse in Software Engineering (RiSE) group and the Chief Scientist at C.E.S.A.R. (the Recife Center for Advanced Studies and Systems). Silvio Lemos Meira is (co-) author of well over 150 referenced publications in software engineering conferences and journals.