A Systematic Mapping Study on Legacy System Modernization

Everton de Vargas Agilar
Computer Centre
University of Brasília
Brasília, Brazil
evertonagilar@unb.br

Rodrigo Bonifácio de Almeida
Computer Science Department
University of Brasília
Brasília, Brazil
rbonifacio@unb.br

Edna Dias Canedo
Faculty of Gama
University of Brasília
Brasília, Brazil
ednacanedo@unb.br

Abstract

Legacy system modernization has gained increasing attention from both researchers and practitioners, mainly due to the need of maintaining legacy systems towards business needs and technology advances. In this way, a set of techniques, tools and terms related to software modernization have been proposed— although they have not been consolidated yet. This hinders the characterization of real modernization scenarios according to the existing literature. This paper synthesize the existing contributions related to software modernization by means of a mapping study that characterizes the main results in terms of proposed processes, techniques, and tools. As one of our main findings, we report a lack of empirical studies trying to understand the benefits of using the existing approaches for software modernization

Keywords Legacy Systems; Software Modernization; Mapping Studies in Software Engineering.

1 Introduction

Modernizing legacy systems takes place when traditional maintenance practices no longer meet the needs of the organizations [1, 3]. In such a scenario, the basic goal is to cut maintenance costs, to turn the legacy systems more flexible to change, and to prolong their usage in a production environment. From the standpoint of organizations, legacy systems correspond to the applications that support business operations in an institution and consolidate most of the corporate data [3].

In spite of being a theme that attracts growing attention, both in the academy and industry, we still lack a summarization of the main research contributions related to the modernization of legacy systems. That is, with the aim of adequately describing real modernization scenarios in a specific institution, we realized the need to conduct a Mapping Study (MS) to characterize the modernization of legacy systems in the context of software maintenance— for the reason that a MS helps researchers to review and consolidate results from studies on a given subject [7, 9]. Therefore, the main contribution of this paper is to characterize software modernization according to the existing literature, discussing the related terms, classifying the related research contributions, and presenting the main reasons that motivate an effort of software modernization (also according to the literature).

Based on the results of our analysis, we found that most of the research contributions to the area are related to the managerial aspects of software modernization (55.88% of the total publications). Accordingly, there existis a lack of research contributions describing (and validating) techniques and tools to support software modernization. The remainder of the paper is organized as follows. Section 2 explains the research method and protocol we adopt to conduct the mapping study. Section 3 describes the main results of the mapping study, by characterizing the research contributions in the field. We conclude the paper presenting some final remarks in Section 4.

2 Research Method

The execution of a MS in Software Engineering has become a established practice that involves a well-defined set of activities [9]. This section describes the protocol used, according to the existing recommendations about how to conduct this kind of research in software engineering. The MS protocol is a plan that contains the basic procedures that should be used in the MS [9], which favors the reproduction of the mapping study by other researchers and diminishes the risk of bias as mentioned in [7]. The remainder of this section presents the research questions, the search strategy, and the criteria for including and excluding the publications.

DOI reference number: 10.18293/SEKE2016-059

2.1 Research Questions

The research questions aim at characterizing the modernization of legacy systems in the domain of software maintenance, by identifying the main contributions and studies found in the literature on the subject. The questions are as follows

(RQ1) What characterizes the modernization of legacy systems according to the existing literature?

(RQ2) What processes, techniques, and tools have been suggested in the literature to support modernization activities of legacy systems?

(RQ3) What are the reasons that lead organizations to modernize their legacy systems?

2.2 Search Strategy

The search strategy consisted of a manual activity surveying publications provided in the main conferences and journals of the Software Engineering research area. This strategy, referenced in [7], was adopted because the terms related to software modernization have not been well defined yet, and thus this manual strategy would allow us to find relevant articles that might be ignored in the case we used an approach based on search strings in digital libraries.

Accordingly, our search strategy was organized to be run in three stages. A list with the research sources was produced for each stage in an empirical way. This strategy was supplemented by the "snowball technique" [7], aimed at finding new primary research sources through the analysis of the references of the articles we found. The research sources selected were:

(a) Research sources in stage 1

- ICSE – Intl. Conf. on Software Engineering
- TSE – Transactions on Software Engineering
- SPE – Software: Practice and Experience
- IEEE Software

(b) Research sources in stage 2

- ICSM – Intl. Conf. on Software Maintenance
- WCRE – Working Conf. on Reverse Engineering
- CSMR – Software Evolution Week

(c) Research sources in stage 3

- ACM Digital Library
- IEEE Xplore
- SpringerLink
- SEI Digital Library
- Science Direct

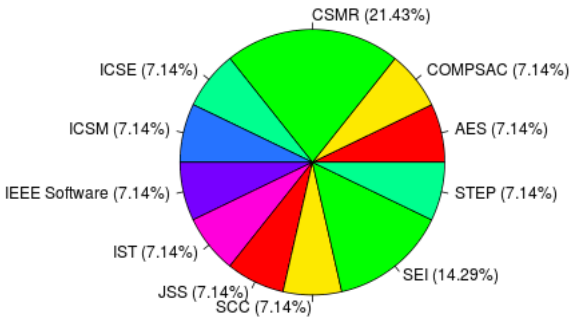

Figure 1. Publications by research sources

2.3 Inclusion and Exclusion Criteria

In order to select the more relevant primary studies, restrictions were set in place for inclusion and exclusion. As regards the inclusion criterion, we only considered publications that alluded to the software modernization theme either in the publication title or in the abstract, and works whose publication date fell between 1995 and 2015. This interval was set in place to yield the highest possible number of relevant publications. We excluded short papers (less than 4 pages in length) and works with less than 20 citations according to Google Scholar.

2.4 Screening of Publications

The selection procedure started with a manual search of the primary research sources that had been previously selected, according to the research protocol. This led to an initial list of 59 publications. This list was then reduced to 44 entries, following the use of a screening technique, as suggested by [9], which discards some publications that did not fit the criteria of the protocol. The final list of publications selected for our analysis can be found at the link `http://goo.gl/WwrGlY`.

Figure 1 summarizes this distribution, grouping by the main conferences and journals that published works related to the subject. The European Conference on Software Maintenance and Re-Engineering (CSMR) is responsible for the highest number of contributions (21.43%), followed by technical reports from the Software Engineering Institute (14.29%).

3 Results

This section describes the MS results obtained after the assessment of the selected publications. As regards the first question, we first analyzed the primary studies with the aim at characterizing *software modernization*. This aspect was then combined to produce the answer to the second question

(RQ2). Finally, w.r.t RQ3 we tackled the reasons that lead the organizations to modernize legacy systems, according to the surveyed literature.

3.1 Analysis of the First Research Question

To answer our first research question, an attempt was made to characterize the modernization of legacy systems in the domain of software maintenance. Therefore, as discussed in [1, 2, 3], modernization can be defined as *the evolution of systems towards new business requirements of the organizations, involving new functionalities, error correction, or technological updates*. In this sense, many theories have been suggested in the literature, as discussed below.

N. Weiderman et al. introduced a model for a software life cycle to describe the evolution of a production system [11]. According to this perspective, there are three distinct stages: maintenance, modernization, and replacement. Small modifications are made during the maintenance stage of a system, through small changes that aims at complying the system with new requirements or bug fixes. The changes with the greatest impact, such as important business requirements, changes in system architecture, or migration of a system to another platform, are done in the modernization stage. However, when the system becomes very resistant to evolution for some specific reason, it must be replaced. In this stage, the business needs of the organization are close related with the implementation efforts to meet these needs. Apart from introducing a life cycle model, Weiderman et al. also propose two approaches for software modernization: (a) white-box for understanding the internal structures of systems and (b) black-box for understanding the external interfaces of legacy systems.

K. Bennett et al. propose a model, entitled staged model, to also describe a system life cycle that assist to identify the main areas of the research on software modernization [1]. This model has 5 stages: initial development, evolution, servicing, phase-out, and close-down. Here, the concept of modernization entails the evolution stage and, differently to the model proposed by Weiderman et al., it is considered a maintenance activity, that can be further classified into 4 classes: adaptive, when there are changes in the software environment; perfective, for new user requirements; corrective, for bug fixes; and preventive, to avoid future problems.

J. Bisbal et al. propose a life cycle model that focuses on the evolutive activities as structured by the impact caused on the systems [3]. Thus, they are divided in wrapping, aimed at providing a new interface for the system's components, making them more accessible to other components; maintenance, for small adjustments and error correction; migration, aimed at moving the legacy system to a more flexible settings, though keeping the original data and functionalities; and re-development, to completely re-write the applications.

It is possible to realize that, although these models use different terms to describe the life cycle of a system, they have many similarities. For instance, the meaning of *replacement* [11] is the same as *re-development* [3] and the meaning of *migration* [3] is equivalent to that of software *modernization* [11, 12]. In addition, the *wrapping stage* described by Bisbal et al. is similar to the *black-box modernization* technique according to Weiderman et al [11, 12].

In continuing with this appraisal, and due to the diversity of terms to describe the approaches for modernization, we answer the other research questions using the model and terminology proposed by Weiderman et al [11, 12]. Accordingly, we briefly introduce each stage in this evolutive model as follows

- **Maintenance** is the first stage in the life cycle of a system. It starts as soon as the system enter into production, being considered an iterative and incremental process through which small modifications are made in the system in a more localised way [1, 12]. However, as discussed in [11], these modifications only meet the needs of an organization for a certain period of time and eventually deteriorate the architecture of the systems.

- **Modernization** takes place when *maintenance* is not enough to keep the system up to date and aligned with the business goals. According to [1, 3, 12], modernization entails more significant changes, such as implementing a novel and relevant functional requirement, a modification on the software architecture, or a system migration to a new software platform. Therefore, as pointed in [1], modernization is more pervasive than maintenance, and this is one of the main aspects in their difference. Finally, as pointed out by [11], the work for modernization should preserve the data and the functionalities of a system, as it would otherwise be characterized as a replacement.

- **Replacement** (also known as *Big Bang* [3]) occurs usually when a legacy system becomes too resistant or inflexible to the work of modernization, there is a lack of documentation, or the cost of software maintenance can no longer be justified [1, 3, 12].

With this brief summary of the features of each stage in the life cycle of a system, the word-cloud of Figure 2 presents the 30 terms most cited in the abstracts of the primary sources selected. It should be noted that, from a technological perspective, there is a certain degree of interest on service-oriented computing in this figure.

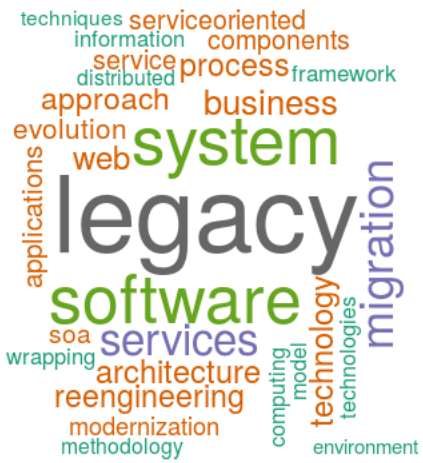

Figure 2. Terms most frequently cited in selected publications

3.2 Analysis of the Second Research Question

To carry out the analysis and answer the second research question (RQ2), we split the publications into three groups, according to the classification proposed in [9]. First, regarding the *focus areas*, we identified three recurrent modernization strategies: black-box modernization, white-box modernization, and replacement. In this group we characterized all publications that propose a new modernization strategy.

We also classified the works by *contribution type*. According to this dimension, a contribution might be (a) *managerial*, for primary sources that describe some process, method, or methodology to manage the modernization process; (b) *technical*, for publications that propose a tool-based solution, such as frameworks, software libraries, and service buses, amongst others; and (c) *management and technical*, in the cases where the work describes a managerial and technical contribution. Finally, we classified the published works by considering the *research type* (as discussed in [13]).

Note in Figure 3 that 63.64% of the publications correspond to *solution proposal*, which suggests a lack of empirical studies trying to understand the benefits of using the existing approaches for software modernization. After concluding the classification of the primary sources, we then generated a bubble-plot reporting the frequencies and distributions of the approaches to modernize legacy systems as identified in the literature. Figure 4 presents that resulting distribution, synthesizing the studies identified in the literature on the subject, along with the gaps and opportunities for future research.

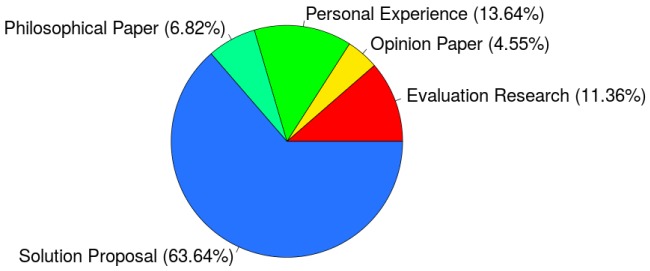

Figure 3. Types of research as reflected in the publications.

It is also possible to see that over 70% of the research work relates to the managerial aspects of the modernization activities. This scenario, according to [10], might be due to the fact that modernization projects have to be aligned with business, organizational, and technical perspectives. In this sense, a process of modernization can assist with several factors, such as: deciding whether developers should (a) continue to maintain the systems (since the costs still justify it); (b) proceed as a modernization effort (which might the best option instead); or (c) conduct a replacement task when it is the only alternative.

Apart from that, when an organization has already decided that modernization is the only way to keep a competitive edge, one needs to decide which *strategies and techniques* will be the most appropriate for each situation. This way, as explained by [8], the decision on how a modernization project should be conducted requires a well-defined process, including modernization strategies, good practices, and the recommendations for an effective management of the project towards modernization. Several works have been proposed in this sense.

For example, Ransom et al. proposes a method for assisting in *system comprehension* [10] that should be undertaken as first activity of a modernization project. This method provides a guide to obtain the necessary information to understand a system, and thus allowing the architects to select a modernization strategy. Moreover, as regards the approaches used, 73.52% resorts to *white-box techniques*, with 23.52% proposing *black-box* ones. This suggests that reverse engineering techniques are promising and have been used to gain a grasp of the systems and to create tools to assist in the modernization process [5, 6, 10].

3.3 Analysis of the Third Research Question

According to the publications considered, it was possible to realize that there are four main reasons for modernizing a legacy systems, as described in the remaining of this section.

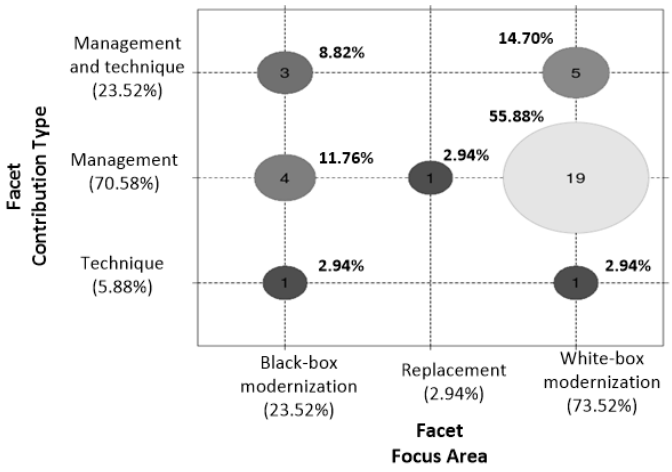

Figure 4. View of the studies identified

3.3.1 Lack of integration between systems

The demand for integrating existing systems is a growing factor in organizations. Software integration allows the automation of business process with improved resource management. However, according to [3, 5], many legacy systems have not been designed to facilitate software integration. This characteristic is considered one of the main reasons that motivates organizations to carry out a modernization effort, particularly towards the integration of business processes. In addition, there are several other benefits obtained as a result of a system integration effort, such as a reduction of duplicate implementation of business rules, the re-use of already-developed software solutions, and the reduction of development costs.

3.3.2 Reducing the maintenance costs

Reducing the maintenance costs of legacy systems is one of the major barriers to that organizations must overcome. According to [1, 3], legacy systems are those that are usually critical for the business and that present unjustifiable costs to be kept in operation. In [4], it is explained that the work of maintenance often monopolizes the efforts made by the organizations, as these activities, including error correction, adaptations, and general improvements consume from 50% to 70% of the budget related to a software effort. In addition, it has been pointed out that the lack of documentation and internal knowledge of the systems is one of the reasons for increasing software development costs, as well as the time spent in maintenance to correct failures in the software [3]. Therefore, as stated by [1], the dilemma faced is that, on one hand, the system is too valuable and a replacement can be too expensive to be considered. On the other hand, keeping a legacy system *up to date* might be too expensive, and thus it might be harder for an organization to evolve the legacy systems so that they can fulfill the the business needs.

3.3.3 Lack of knowledge

As mentioned earlier, the lack of knowledge and the lack of legacy systems understanding is one of the reasons for a modernization project. According to [1, 3], understanding the design of a system is regarded as one of the requirements to implement the changes necessary by the organizations. It has been reported in the literature that a substantial part of the time needed to understand a legacy system lies in locating domain concepts in source code [1]. Thus, understanding the legacy systems is one of the central research problems in the literature, as discussed in [1]. For this reason, several research works have been proposed to identify alternatives for gaining a better understanding of the system, a vital component of any effort towards software evolution [1, 10].

3.3.4 Error proneness

Bennet also arguments that, due to the lack of updated documentation, modernization efforts are often made considering the source code as a reliable documentation [1]. Along with the issues in staff management, the systems can be affected by a lack of system and domain knowledge.

3.4 Threats to Validity

This study is limited to research in the literature to characterize the modernization of legacy systems in the context of maintenance software. Thus, the main threats to validity of this research is some possible bias in the procedures for selecting the publications. The research protocol consisted of manual searches in conferences and journals of Software Engineering—instead of using a search string, which is often applied in a MS. We believe that this decision helped us to obtain the most relevant articles for this study, in particular because the terms used to refer to *software modernization or evolution* of legacy systems are very wide (as discussed in Section 3.1). Our criteria led to 44 contributions published between 1995 and 2015, with at least 4 pages long and 20 citations according to Google Scholar. We also believe that this study can be reproduced by other researchers without the problems of publication bias. Of course, other studies might comprise different goals and research questions, and might also be more comprehensive. However, results or trends identified in this study should remain the same for the investigated period.

4 Final Remarks

Modernization of legacy systems has gained much attention in the last years, leading to a number of research contributions presenting new methods, techniques, and tools. Nevertheless, the lack of a suitable consolidation of these results hinders both researchers and practitioners to conduct their activities as well as to describe their findings and experiences using a common knowledge. In this paper we presented the results of a mapping study (MS) that consolidates the main contributions to the field. We found that the majority of the publications relies on a kind white-box modernization approach (often recovering the necessary information of legacy systems from the source code), which reinforces the need for reverse engineering tools. In addition, we also find that the managerial aspects are most relevant, which reinforces the idea of the importance of a good strategy of modernization. However, we found just a few studies reporting success experiences in modernizing legacy systems. Actually, most of the publications detail solutions that have been proposed without any practical evaluation. Finally, we found that there are four recurrent reasons reported in the literature to modernize a legacy system: the need for legacy systems integration, the need for improving software flexibility, the lack of knowledge about the system, and the error proneness for maintaining an existing system.

5 Acknowledgments

We would like to thank FAPDF Brazilian research funding agency for partially supporting this work.

References

[1] BENNETT, K. Legacy systems: coping with success. *Software, IEEE 12*, 1 (1995), 19–23.

[2] BIANCHI, A., CAIVANO, D., MARENGO, V., AND VISAGGIO, G. Iterative reengineering of legacy systems. *Software Engineering, IEEE Transactions on 29*, 3 (2003), 225–241.

[3] BISBAL, J., LAWLESS, D., WU, B., AND GRIMSON, J. Legacy information systems: issues and directions. *IEEE Software 16*, 5 (Sep 1999), 103–111.

[4] CANFORA, G., CIMITILE, A., DE LUCIA, A., AND DI LUCCA, G. A. Decomposing legacy programs: A first step towards migrating to client–server platforms. *Journal of Systems and Software 54*, 2 (2000), 99–110.

[5] CHUNG, S., AN, J., AND DAVALOS, S. Service-oriented software reengineering: Sosr. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (Jan 2007), pp. 172c–172c.

[6] FLEUREY, F., BRETON, E., BAUDRY, B., NICOLAS, A., AND JÉZÉQUEL, J.-M. Model-driven engineering for software migration in a large industrial context. In *Model Driven Engineering Languages and Systems*. Springer, 2007, pp. 482–497.

[7] KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University 33*, 2004 (2004), 1–26.

[8] LEWIS, G., MORRIS, E., AND SMITH, D. Service-oriented migration and reuse technique (smart). In *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on* (2005), pp. 222–229.

[9] PETERSEN, K., FELDT, R., MUJTABA, S., AND MATTSSON, M. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* (Swinton, UK, UK, 2008), EASE'08, British Computer Society, pp. 68–77.

[10] RANSOM, J., SOMERVILLE, I., AND WARREN, I. A method for assessing legacy systems for evolution. In *Software Maintenance and Reengineering, 1998. Proceedings of the Second Euromicro Conference on* (1998), IEEE, pp. 128–134.

[11] SEACORD, R. C., PLAKOSH, D., AND LEWIS, G. A. *Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[12] WEIDERMAN, N., SMITH, D., AND TILLEY, S. Approaches to legacy system evolution. Tech. Rep. CMU/SEI-97-TR-014, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1997.

[13] WIERINGA, R., MAIDEN, N., MEAD, N., AND ROLLAND, C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering 11*, 1 (2006), 102–107.