INVESTIGACIÓN e INNOVACIÓN en INGENIERÍAS
ISSN2344-8652

Factores de éxito y barreras de adopción en la reutilización de software: una revisión de la literatura

Success factors and adoption barriers in software reuse: a literature review

Luisa Fernanda Restrepo Gutierrez **Elizabeth Suescún Monsalve**

Raúl Mazo **Daniel Correa**

Paola Vallejo

Universidad EAFIT, Colombia

OPEN ACCESS

Recibido: 22/09/2021
Aceptado: 22/10/2021
Publicado: 13/12/2021

Correspondencia de autores:
lrestr61@eafit.edu.co

Resumen

Objetivo: Conocer el estado del arte de la reutilización de software, centrándose en sus factores de éxito y barreras de adopción. **Metodología:** Para alcanzarlo realizamos una revisión de la literatura en la cual adoptamos algunos elementos de un estudio de mapeo sistemático. Esta revisión de la literatura constó de cinco etapas (i) definición de las preguntas de investigación, (ii) definición de la estrategia de búsqueda, (iii) definición de los criterios de inclusión/exclusión, (iv) realización de la búsqueda y, finalmente, (v) resolución de las preguntas. **Resultados:** El trabajo recopila veintiséis factores que influyen en la reutilización del software y las barreras de adopción encontradas en la literatura. Estos factores se clasificaron en cuatro perspectivas: factores organizativos, factores empresariales, factores tecnológicos y factores de proceso. **Conclusiones:** Los factores resultantes podrían utilizarse para introducir prácticas de reutilización de software y conocer el estado actual de la reutilización de software en las empresas. Asimismo, los factores resultantes podrían aplicarse en la industria del software para conocer su estado actual de práctica.

Palabras clave: Reutilización de software, ingeniería de software, estudio de la literatura, prácticas de éxito, barreras de adopción.

Abstract

Objective: This research aims to study the state of the art of software reuse, focused on success factors and adoption barriers. **Methodology:** To reach the objective, a literature review was conducted in which we adopted some elements from a Systematic Mapping Study (SMS). This literature review was divided into five stages: (i) definition of research questions, (ii) definition of the search strategy, (iii) definition of the inclusion and exclusion criteria, (iv) search conduction, and finally, (v) resolution of the research questions. **Results:** The paper compiles 26 factors that influence software reuse and common adoption barriers found in the literature. These factors are generally classified into four perspectives: organizational factors, business factors, technological factors, and process factors. **Conclusions:** The resulting factors could be used to introduce software reuse practices and understand the state of the art of software reuse in companies. In addition, resulting factors could be used in the software industry to know its current state of the practice.

Keywords: Software reuse, software engineering, literature review, success practices, adoption barriers.

Como citar (IEEE): L. Restrepo-Gutierrez., E. Suescún-Monsalve., R. Mazo., D. Correa., y P. Vallejo. "Factores de éxito y barreras de adopción en la reutilización de software: Una revisión de la literatura". Investigación e Innovación en Ingenierías, vol. 9, n°3, 93-107, 2021. DOI: https://doi.org/10.17081/invinno.9.3.5565

Introduction

Software reuse is the process of creating software systems from existing software assets instead of building all software assets from scratch [1]. In plain language, reuse is based on the principle of “not reinventing the wheel”. In software development, reuse does not only concern source code, however it also includes all the assets produced in the software development life cycle (e.g., requirements, architectures, test cases, knowledge, and documents). The claim that the systematic reuse of software provides significant benefits to the industry has been widely accepted among researchers [2, 3, 4, 5]. Some of these benefits are, for instance, increased productivity and quality, reduced development time and costs, reduced effort in documentation and maintenance, and improved time-to-market of software products.

Software reuse takes different forms varying from ad-hoc reuse to systematic reuse [6]. The ad-hoc reuse is characterized by the lack of procedures and is opposed to systematic reuse, where software reuse is planned [7]. The effective implementation of software reuse faces technical and non-technical inhibitors [8, 9] such as lack of comprehensively reuse-oriented CASE tools, lack of model-level component libraries ready for use across platforms, immaturity of methodologies that integrate multiple reuse techniques, immaturity of standard languages to support model-level reuse, immaturity of a distributed architecture for the publication of third-party components, difficulties in locating and accessing components, fierce competition from the software industry, leading to an obsession with time-to-market of next product or product release, reuse ROI (Return On Investment) is only in the long term, cost of delay almost impossible to estimate accurately, reuse requires costly software house organization re-engineering, changing team structures, productivity measures oriented to the number of lines of source code written, design for reuse and design from existing assets heavily under-taught, developers with hard-to-find reuse skills, “Not invented here" syndrome [10], hacker mentality, resistance to change habits and lack of business models for component asset suppliers [8,11, 12, 13, 14]. This paper presents a literature review in which we adopt some elements from a Systematic Mapping Study (SMS) to determine which factors influence the success of software reuse.

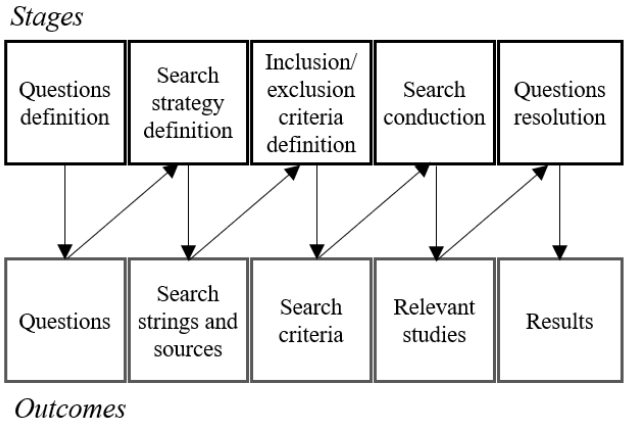
This article is organized as follows. Section 2 presents the search methodology and selection of relevant articles. Section 3 gathers the articles found in the literature and discusses the results of the articles reviewed. Finally, Section 4 presents the conclusions of this research and discusses possible further work.

Methodology

The search strategy and the search process used to find the relevant articles are explained below.

This research applied some stages of the Systematic Mapping Study (SMS) recommendations proposed by Petersen et al. [15] An SMS is a “form of a secondary study aiming to provide a comprehensive overview of a certain research topic, to identify gaps in the research, and to collect evidence to guide researchers and practitioners in their current or future work" [16] and Zhang et al. [17] describe the importance of this kind of studies in the software engineering area. It allows to analyze all available studies in a domain at a high level; therefore, it will allow to answer broad research questions about the current state of the research on a topic [18]. In particular, our literature review process has five stages: (see Fig. 1): (i) questions definition, (ii) search strategy definition, (iii) inclusion and exclusion criteria definition, and (iv) search conduction. Each stage produces an outcome used as input to the next stage.

Figure 1: Literature review process, taken and adapted from Petersen et al. [10]



Therefore, it implies the presentation of the design of the experiment used in the study, the instruments for data collection, the participants, the procedures carried out during the study, the data analysis, and the eventual conflicts of interest.

↠ Questions definition

In the first stage, two answerable and interpretable questions were formulated to identify the state of the art of the research, specifically the key factors and adoption barriers in software reuse:

- SQ1. What are the key factors for successful software reuse adoption?
- SQ2. What are the adoption barriers in software reuse?

↠ Search strategy

Keeping in mind the previous research questions, the search strings presented in Table 1 were defined. These terms consider two topics: (i) software reuse and research focus and (ii) software reuse and applications. The first topic refers to words and phrases related to the goal of the literature review. The second topic wants to reach out to related studies that do not specifically mention key factors still, they consider practicing or studying it in specific aspects. Thus, each of the two topics complements the other.

We defined two search strings with terms and sentences used in the search process. Finally, we refined (added and adjusted) the search strings with keywords used in relevant studies about our research area. In addition to the search strings, the search sources used to find the primary studies were stated. The search sources included Springer Link, IEEE Xplore, ACM digital library, Science Direct, Scopus, and ISI Web of Science. As some relevant publications may not be referenced or indexed in the previous digital libraries, it was decided to extend the search in Google Scholar and CiteeSeer engines, with the same search strings as above.

Table 1. Definitive search strings

String	Topics	Search string
S1	Software reuse and research focus	("software") AND ("reuse" OR "reusability" OR "asset reuse") AND ("Issue" OR "barrier" OR" failure factor" OR "adoption" OR "obstacle" OR "impediment" OR "trend" OR "practice" OR "success factor" OR "successful practice" OR "successful method" OR "successful process")
S2	Software reuse and applications	("software") AND ("reuse" OR "reusability" OR "asset reuse") AND ("case study" OR "industrial case" OR "social aspect" OR "economics aspect" OR "marketing aspect" OR "organizational aspect" OR "business aspect" OR "process aspect")

⇝ Inclusion/Exclusion criteria definition

For publications to be included or excluded, the following limits and quality criteria were established. These criteria were created in order to find the most relevant papers to get answers to the research questions and avoid papers that do not fit this field and do not allow the research questions to be solved or considered irrelevant to our study.

Inclusion criteria. IC1. Papers, dissertations, case studies, and book chapters that study success practices, adoption barriers, or influence factors in software reuse. IC2. Papers, dissertations, book chapters and case studies published between January 2000 and July 2021. IC3. Papers, case studies, dissertations, and book chapters available in electronic form.

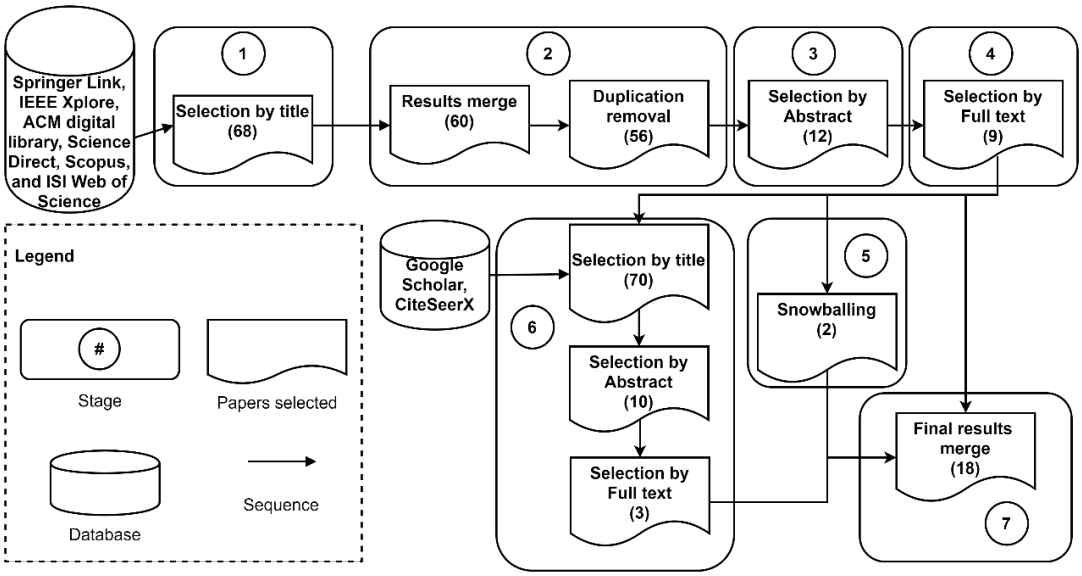
Exclusion criteria. EC1. Studies that describe events, posters, or unpublished works. EC2. Papers that do not focus on success practices, adoption barriers, or influence factors in software reuse. EC3. Papers whose approaches or techniques are not apply to the software reuse context. EC4. Approaches without details on how success practices, adoption barriers, or influence factors in software reuse are obtained.

⇝ Search conduction

Based on the study given by Li *et al.* [19], the search process was divided into seven steps: (i) selection by title, (ii) first results merge, (iii) selection by abstract, (iv) selection by full text, (v) snowballing, (vi) search extension in Google Scholar and CiteeSeer, and (vii) final results merge. The previously defined search strings, search engines, and inclusion/exclusion criteria were used in the process. Each stage is detailed below (see Fig. 2):

- **Selection by title:** this stage started by using the search strings in seven search sources (ACM, IEEE Explore, ScienceDirect, Scopus, Springer Link, and ISI Web of Science), then, depending on the title the potential studies were chosen. In this step, the inclusion criteria IC1, IC2, and IC3 were used.
- **First results merge:** All potential papers were merged (68 studies at this point), but duplication studies were discovered. A duplicated study is one that has been found from many search engines (i.e., digital libraries) because of the overlapping between these sources. The final set of relevant studies must not include duplicate studies. Consequently, duplicated studies will be excluded in the first scanning stage, simply keeping one copy of the document (the most complete or recent version). In the end, eight studies were deleted since they were duplicates.
- **Selection by abstract:** in this stage, the candidate studies' abstracts were analyzed to guarantee that they were related to the desired topic (software reuse factors or adoption barriers); at this point, 12 candidate studies were selected.

- **Selection by full text:** we read and analyzed the full texts selected in stage 3. In this stage, the exclusion criteria EC1, EC2, EC3, and EC4 were used, resulting in the selection of nine studies.
- **Snowballing:** to locate other potentially relevant papers, the snowballing technique [20] was used. Checking the references of the studies chosen in stage 4 is known as snowballing. As new research is discovered, this process may become iterative. Only the first iteration, however, was used, resulting in two new studies found.
- **Search extension in Google Scholar and CiteSeer:** parallel to stage 5, Google Scholar and CiteSeer were used to broaden the scope of the search. We used the search strings and made the first scan by title (In this stage, the inclusion criteria IC1, IC2, and IC3 were used). Then, a second scan by abstract was made, and finally, the third scan by full text (exclusion criteria EC1, EC2, EC3, and EC4 were applied in this stage). Three new studies were found. For Stages 3 and 6, all candidate studies discussed software reuse; however, we were interested in studies that discussed factors for successful software reuse adoption. That is the reason why we only selected a few studies.
- **Final results merge:** this stage merged the selected studies from stages 4, 5, and 6; thus, 18 relevant studies were selected after a detailed analysis. They serve to answer the research questions, and also to present the factors required to carry out this study. The selected studies list is available in [21].



↠ Threats to validity

Although this article aims not to present an SMS, the fact that it used some stages of a protocol to conduct a literature review makes it relevant to consider threats to validity. The following are threats to the validity of our literature review, (i) the search strings in the searching process may include inadequate search terms related to the research topic, (ii) only automatic search was applied, and (iii) it was selected papers that had a general discussion about software reuse success factors from the first half of 2000 to 2021. All these factors can bias the identification of primary studies [22]. Since the subject has been explored for 30 years, it does not guarantee all related primary studies and reuse factors were selected. These threats to validity may have an impact on the generalizability of our literature review results. To minimize these threats and avoid bias in data extraction, all the process was executed through a cross-check by the authors.

Results

After analyzing the resulting papers, the literature review questions (SQ1 and SQ2) were resolved; the results are presented below.

Table 2. Success practices and influence factors for software reuse

Type	#	Success practices and influence factors	References
Business	1	Application domain	[23, 24, 25]
	2	Type of software developed	[26]
	3	Product family approach	[23, 25], [27, 28, 29]
	4	Domain analysis	[25, 29, 30]
Organizational	5	Software reuse education	[25, 28, 31]
	6	Management commitment	[25, 26, 30, 32]
	7	Project team experience	[31, 32, 33]
	8	Reward and incentives	[23, 24, 25, 33]
	9	Economic feasibility	[24, 25, 28, 31]
	10	Legal issues	[25, 28]
	11	Software organization and team size	[26, 31]
	12	Independent reusable assets development team	[23, 25, 28]
Process	13	Specific function in the software reuse process	[29, 34]
	14	Origin of the reused assets	[23, 24, 28, 29, 31]
	15	Configuration management	[23, 25. 29, 32]
	16	Software reuse measurement	[25, 28, 29, 31]
	17	Quality model's usage	[23, 25, 28, 35]
	18	Reused assets type	[23, 24, 27]
	19	Previous development of reusable assets	[25, 28, 29]
	20	Development of assets for reuse	[31]
	21	Software certification process	[25, 30]
	22	Systematic reuse process	[23, 24, 25, 28, 31, 32]
Technological	23	Repository systems usage	[25, 28, 29, 30, 33]
	24	CASE tools usage	[23, 25, 28, 29]
	25	Software development approach	[23]
	26	Programming language	[23, 24, 25, 26, 31]

↠ Key factors for software reuse

Some authors define these elements as "key factors", "success factors", "factors of predictive importance" or "influence factors", among others [12, 23, 35, 36, 37, 38, 39, 40,41, 42], [24, 25, 27, 28, 30, 31, 32, 33]. Some of these factors were selected and used to answer SQ1. These factors were grouped into four categories: business, process, organizational, and technology. These categories were based on categories defined by Lucrédio et al.'s study [23]. Table 2 presents a brief description of these key factors.

Business Factors

Business factor or "domain focus" is an indicator for the level of among products' commonalities Developing applications for a specific domain will probably result in high commonalities among created solutions. Therefore, the software reuse levels are influenced by an organization's strategic choice to focus on a particular domain [29]. The following were the factors found for this category:

- **Application domain.** The application domain is where the developed software system will be used; it determines how a software project will be oriented and executed [43]. Thus, the application domain is an essential factor to consider; according to Lucrédio et al. [23], it seems that some application domains, such as financial, education, and manufacturing have more success with software reuse than other domains.
- **Type of software system.** This factor focuses on the software system types; for instance, web applications, embedded software, database systems, etc. [26].
- **Product family approach.** This factor focuses on building software products around a core set of reusable software assets [23, 25, 27, 28, 29]. With this approach, product quality is improved, time-to-market is reduced, and productivity is increased.
- **Domain analysis**. "Domain analysis is a generalization of systems analysis. The primary objective is to identify the operations and objects needed to specify information processing in a particular application domain" [44]. This factor identifies domains and software assets that are good candidates for reuse [28, 30, 40, 44].

Organizational Factors

Reuse is not just a technical issue, but also a people issue. Therefore, successful reuse programs must be integrated within the company's culture. Because organizational factors can significantly affect the implementation of reuse programs [13, 45]. The following were the factors found for this category:

- **Software reuse education.** This factor emphasizes that the organizations should encourage and provide the employees with formal and informal training sessions about software reuse. In addition, organizations must shift their thinking and establish a culture of software reuse [20, 23, 31].
- **Management commitment.** This factor emphasizes that the organization's top managers must be involved in the software reuse culture. They should demonstrate strong support by allocating funds [18] and should understand reuse issues and benefits for the company [30, 33, 40].
- **Project team experience.** Project team experience is the time or experience level of the team. Team experience can influence the success of software reuse. Teams with more experienced developers would obtain more success since they have more domain knowledge than the knowledge of less experienced teams [23].
- **Reward and incentives.** Incentives are stimuli offered by the company to increase production and improve performance. Incentives are important to motivate developers to apply systematic software reuse during the initial stages because additional efforts are needed. Incentives increase commitment to goal attainment [9, 23, 33].
- **Economic feasibility.** This factor focuses on whether a business or a project is feasible cost-wise and logistically. Companies should analyze whether software reuse will have the proper kind of return that warrants the up-front investment in reusability [24, 28, 31].
- **Legal issues.** This factor focuses on legal restrictions, such as contracting rules, ownership, and liability. These restrictions can impede successful software reuse. In addition, the following types of protection should also be considered: trade secret protection, patent protection, and copyright protection [28, 46].
- **Software organization and team size.** This factor refers to the number of employees in an organization and a software team. According to Morisio [26] and Frakes [31], the following elements can influence software reuse: team size, production strategy, maturity level, and project size.

- **Independent reusable assets development team.** This factor focuses on the creation of an independent team which is specialized in software reuse, and which builds more robust assets. This minimizes the effort to reuse these assets [23, 47].

Process Factors

Implementing a reuse approach requires that the company designs a suitable process. The company should generally define reuse roles and responsibilities, the company should add reuse processes, the company should modify non-reuse processes, and install tools [34]. The following were the factors found for this category:

- **Specific function in the software reuse process.** Roles clarify who develops a company's reusable assets and when they are developed [34]. This factor focuses on the specification of reuse roles as part of the company's organizational structure. For example, reusable asset producers should be separated from reusable assets consumers [29].
- **Origin of the reused assets.** This factor focuses on the source of their reused software assets. The companies must know if these assets are developed internally or externally. Instead of developing an asset from scratch, an organization can reuse existing Commercial-Off-The-Shelf (COTS) or internet distributed software assets. Another option is to build assets from existing products through an re-engineering process, or adapting existing products with minimum effort [23, 28, 29, 31].
- **Configuration management.** This factor refers to keeping a record of the assets used in all the company projects, to control quality during the evolution of the assets [23, 29, 32].
- **Software reuse measurement.** This factor focuses on information about asset reuse. Companies should collect information such as lines of code, the number of files, or specific metrics. This information helps to identify the most reused assets, unused assets, assets coupling, and quality, among others [29, 30, 31].
- **Quality model's usage.** This factor refers to the quality and maturity models of the companies. These models define a quality process that can be tested to guarantee the quality of the final products [23, 28, 35].
- **Reused assets type.** This factor focuses on the type of reused assets. Several assets can be reused, such as specifications, design, documentation, test cases, data, etc. [27], not only source code as many organizations believe [23].
- **Previous development of reusable assets**. Reuse planning describes whether reuse events are planned at the beginning of a software development project or during the development process [28, 29]. Assets to create can be small or large, including the design and requirements of these assets for which a requirements management process is necessary. The assets can be developed from scratch or re-engineered and should be created and maintained by a specific group.
- **Development of assets for reuse.** This factor focuses on the type of assets developed to be reusable in future projects. Companies should anticipate which assets will be reused, there should be a strategy to raise awareness for software developers and managers to reuse those assets [31, 48].
- **Software certification process.** This factor focuses on the certification processes of the company. Certifying reusable software assets increases their reliability. Assets should meet specific metrics and should reach the required quality standards. The reuse of these certified assets decreases the failure rate [30].

- **Systematic reuse process.** This factor refers to the software reuse process followed by software developers and maintainers; this process is defined and integrated with the software development process defined by the organization. It is a key business strategy to improve product quality [23, 24, 28, 31, 32].

Technological Factors

Technological factors refer to the means for building and integrating the coding assets [40]. “Proper mechanisms are needed to ensure that guidelines, techniques, and standards for making things reusable are developed and followed" [33]. The following were the factors found for this category:

- **Repository systems usage.** This factor focuses on the software repositories used to place reusable assets. Software repositories allow developers to search, comment, and track software assets [28, 29, 30, 33].
- **CASE (Computer-Aided Software Engineering) tools usage.** This factor focuses on the CASE tools used. Companies may select and use tools that support the work in progress related to software reuse, implementation, testing, and management of software assets. Managers need tools to evaluate metrics, project management, and communication, and thus, describe the active and effective support of communication among producers and consumers of assets [23, 28, 29].
- **Software development approach.** This factor refers to the programming paradigm or software approach used in the company. Selecting a programming paradigm or software approach to developing the company’s reusable assets is a critical decision that impacts assembling and reusability [23].
- **Programming language.** This factor refers to the programming languages used by companies to develop software systems. For example, Frakes [31], Lucrédio [23], Morisio [26], and Rafael [24] consider in their investigations the programming language as a factor impacting reuse.

↠ Adoption barriers for software reuse

Adoption barriers represent elements that affect and influence the adoption of software reuse. The barriers presented in Table 3 respond to question SQ2. The table summarizes the adoption barriers reported by Sherif and Vinze [49], Bass et al. [36], Jha et al. [41], and Keswani et al. [50]. The adoption barriers were classified according to the Sherif and Vinze [9] proposed categories: Top management category represents investments in time and resources to develop reuse assets, measures the benefits and cost based on quantitative terms, and the companies’ economic models. The software developer’s category groups the developers’ feelings and perspectives about trust of the original asset producer and strategies for identifying reuse opportunities, building reusable and quality assets. The organizational category represents the organization of the reuse group as well as its performance evaluation and the endeavor to resolve conflicts or disagreement and mitigate risk factors for software development. The technical category is attribute to the complexity of the reuse technology; specifically, the process included in building reusable assets, the reusable assets themselves, and all that support needed for using and maintaining the assets.

Table 3. Adoption barriers for software reuse

Categories	Barrier	References
Top management	Lack of a source of funding	[49]
	Initial economic investment	[49, 50]
	Lack of top management commitment	[41]
Software developers	Mistrust of reusable artifacts	[50]
	Software developer's resistance	[50]
	Lack of skills	[50]
	Resistance to systematic reuse	[49]
Organizational	Lack of quantitative measures to assess the benefits and costs	[49]
	Lack of an autonomous reuse group	[49]
	Lack of incentives	[12, 49]
	Lack of reuse policies to govern the creation of reusable assets	[49]
	Lack of software quality	[49]
	Lack of an educational and training program or a mentoring	[41, 49]
	Large size of the industry	[50]
Technical	Lack of adoption of existing requirements reuse proposals	[42]
	Lack of available components	[36]
	Lack of stable standards for component technology	[36]
	Lack of certified components	[36]
	Lack of a technology that supports reuse	[49]
	Lack of complete and reliable components	[36, 41]
	Lack of information on the availability of reusable assets	[49]
	Lack of assets management	[49]
	Incompatibility with traditional methodology	[49]
	Difficulty to locate reusable assets	[49]
	Lack of a well-organized and indexed software repository	[41, 49]

Discussion

Twenty-six factors associated with software reuse were identified in the literature. Besides, twenty-four adoption barriers were identified. These factors are generally divided into four perspectives: organizational factors, business factors, technological factors, and process factors.

Concerning state of the art (see Table 4), a high similarity between the resulting papers was found. Some papers led to the same conclusion of influence on software reuse success such as application domain, product family approach, domain analysis, management commitment, economic feasibility, team size, the origin of the reused assets, configuration management, reused assets type, systematic reuse process, an independent reusable assets development team, the previous development of reusable assets, and development of assets for reuse. Evidencing that they have been conserved over time. However, the remaining factors had different conclusions where it was found that:

From the Frakes and Fox [31], we found that project team experience, reward and incentives, legal issues, software reuse measurement, CASE tools usage, and programming language factors that were not considered influencing now are factors that influence software reuse success, as suggested by most of the results in the literature (see Table 4).

Morisio et al. study in 2002 [26] concluded that software development approach and quality models usage factors do not influence software reuse success. This may indicate that the influence of these factors is

different on the European and Latin America case since context application and the way of developing changes.

For the type of software developed, software reuse education, specific function in the software reuse process, software certification process, and repository systems usage factors more research should be done, because there are inconclusive results, as there are different conclusions for both the Latin American, European, and United States cases.

Given the discussion, the resulting factors and adoption barriers could be applied in the software industry to know its current state of practice. This will allow us to identify the industry's opportunities, trends, and weaknesses in software reuse.

Table 4. Comparison software reuse influence with related works: Yes = some influence on reuse; No = no influence on reuse; N/D = no data.

Paper	[31]	[27]	[30]	[26]	[32]	[28]	[23]	[33]	[29]	[35]	[25]	[24]
1 Application domain	Yes	N/D	N/D	N/D	N/D	N/D	Yes	N/D	N/D	Yes	Yes	Yes
2 Type of software developed	Yes	N/D	N/D	No	N/D	N/D	No	Yes	N/D	N/D	N/D	N/D
3 Product family approach	N/D	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/D
4 Domain analysis	N/D	N/D	Yes	Yes	Yes	Yes	N/D	Yes	Yes	Yes	Yes	N/D
5 Software reuse education	Yes	No	Yes	N/D	N/D	Yes	No	Yes	N/D	Yes	Yes	N/D
6 Management commitment	N/D	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	N/D
7 Project team experience	No	Yes	N/D	N/D	N/D	N/D	Yes	N/D	Yes	Yes	Yes	N/D
8 Reward and incentives	No	N/D	N/D	N/D	N/D	Yes	N/D	Yes	N/D	N/D	Yes	Yes
9 Economic feasibility	Yes	N/D	Yes	N/D	N/D	Yes	N/D	Yes	N/D	Yes	Yes	Yes
10 Legal issues	No	N/D	N/D	N/D	N/D	Yes	N/D	Yes	N/D	N/D	Yes	No
11 Software organization and team size	No	N/D	N/D	No	N/D	N/D	No	N/D	N/D	N/D	N/D	N/D
12 Independent reusable assets development team	N/D	Yes	N/D	N/D	N/D	Yes	Yes	N/D	Yes	N/D	N/D	N/D
13 Specific function in the software reuse process	N/D	N/D	N/D	No	N/D	N/D	N/D	N/D	Yes	N/D	Yes	N/D
14 Origin of the reused assets	N/D	N/D	N/D	N/D	N/D	Yes	Yes	N/D	N/D	N/D	N/D	N/D
15 Configuration management	N/D	N/D	N/D	N/D	N/D	N/D	Yes	N/D	Yes	Yes	N/D	Yes
16 Software reuse measurement	No	N/D	Yes	N/D	Yes	Yes	N/D	Yes	Yes	Yes	Yes	N/D
17 Quality model's usage	Yes	N/D	N/D	No	N/D	Yes	Yes	N/D	N/D	N/D	Yes	N/D
18 Reused assets type	N/D	Yes	N/D	N/D	N/D	N/D	Yes	N/D	N/D	Yes	Yes	N/D
19 Previous development of reusable assets	N/D	N/D	N/D	N/D	Yes	Yes	Yes	Yes	Yes	Yes	N/D	Yes
20 Development of assets for reuse	N/D	Yes	N/D	N/D	N/D	N/D	N/D	N/D	N/D	Yes	Yes	N/D
21 Software certification process	No	No	Yes	N/D	Yes	N/D	N/D	Yes	Yes	Yes	N/D	N/D
22 Systematic reuse process	Yes	N/D	Yes	Yes	Yes	Yes	Yes	Yes	N/D	Yes	Yes	N/D

23 Repository systems usage	No	No	N/D	No	N/D	Yes	No	Yes	Yes	Yes	Yes	Yes
24 CASE tools usage	No	Yes	Yes	N/D	N/D	Yes	Yes	N/D	Yes	Yes	Yes	Yes
25 Software development approach	N/D	N/D	N/D	No	Yes	N/D	Yes	N/D	N/D	N/D	Yes	N/D
26 Programming language	No	N/D	N/D	N/D	N/D	N/D	Yes	N/D	N/D	N/D	N/D	N/D

Conclusions

This paper is a compilation of the factors and adoption barriers found in the literature; twenty-six factors were identified in summary. These factors are generally divided into four perspectives: organizational factors, business factors, technological factors, and process factors. And twenty-four adoption barriers were identified. The resulting factors could be used to introduce software reuse practices, understand state of the art in software reuse, and identify adoption barriers of this critical engineering practice for which companies can be reflected.

In all of our research, one thing has emerged consistently. Since nearly every company relies on software, an exemplary software reuse process leads to enhanced reliability and productivity and could reduce cost. And software reuse is affected by many factors, including leadership, tools, maturity, the culture of continuous learning/education, improvement, and reuse effectiveness vary with company size.

As an opportunity, identified factors and adoption barriers could be used to develop and carry out a survey that will allow to define the state of the practice of software reuse companies and understand how these factors influence and contribute to the successful implementation of software reuse practices.

References

1. C. W. Krueger and C. W., “Software reuse,” ACM Comput. Surv., vol. 24, no. 2, pp. 131–183, Jun. 1992, doi: 10.1145/130844.130856.
2. W. B. Frakes and Kyo Kang, “Software reuse research: status and future,” IEEE Trans. Softw. Eng., vol. 31, no. 7, pp. 529–536, Jul. 2005, doi: 10.1109/TSE.2005.85.
3. J. Barros, F. Benitti, and S. Matalonga, “Trends in software reuse research: A tertiary study,” Comput. Stand. Interfaces, vol. 66, p. 103352, Oct. 2019, doi: 10.1016/J.CSI.2019.04.011.
4. Y. Kim and E. A. Stohr, “Software Reuse: Survey and Research Directions,” J. Manag. Inf. Syst., vol. 14, no. 4, pp. 113–147, 1998, doi: 10.1080/07421222.1998.11518188.
5. P. Mohagheghi and R. Conradi, “Quality, productivity and economic benefits of software reuse: A review of industrial studies,” Empir. Softw. Eng., vol. 12, no. 5, pp. 471–516, Oct. 2007, doi: 10.1007/S10664-007-9040-X/TABLES/16.
6. A. Govardhan and P. Premchand, “A PRAGMATIC APPROACH TO SOFTWARE REUSE,” 2010.
7. A. Dubey and H. Kaur, “Reusability Types and Reuse Metrics: A Survey,” 2015. [Online]. Available: https://pdfs.semanticscholar.org/2904/2ccd2e348422c4767d6920158aedc04fd862.pdf.
8. Kalagiakos, “The non-technical factors of reusability,” in 2003 Proceedings 29th Euromicro Conference, 2003, pp. 124–127, doi: 10.1109/EURMIC.2003.1231577.

9. S. Karma, A. Radha, and L. Zhangxi, “Resources and incentives for the adoption of systematic software reuse,” Int. J. Inf. Manage., vol. 26, no. 1, pp. 70–80, Feb. 2006, doi: 10.1016/J.IJINFOMGT.2005.08.007.
10. A. Stefi, “Do Developers Make Unbiased Decisions? - The Effect of Mindfulness and Not-Invented-Here Bias on the Adoption of Software Components,” undefined, 2015, doi: 10.18151/7217489.
11. B. Xu, L. An, F. Thung, F. Khomh, and D. Lo, “Why reinventing the wheels? An empirical study on library reuse and re-implementation,” Empir. Softw. Eng., vol. 25, no. 1, pp. 755–789, Jan. 2020, doi: 10.1007/s10664-019-09771-0.
12. J. L. Barros-Justo, D. N. Olivieri, and F. Pinciroli, “An exploratory study of the standard reuse practice in a medium sized software development firm,” Comput. Stand. Interfaces, vol. 61, pp. 137–146, 2019, doi: https://doi.org/10.1016/j.csi.2018.06.005.
13. J. Sametinger, Software Engineering with Reusable Components. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.
14. J. Margono and T. E. Rhoads, “Software reuse economics,” in Proceedings of the 14th international conference on Software engineering - ICSE ’92, 1992, pp. 338–348, doi: 10.1145/143062.143151.
15. K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic Mapping Studies in Software Engineering,” in Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, 2008, pp. 68–77.
16. C. Wohlin, P. Runeson, P. A. Da Mota Silveira Neto, E. Engström, I. Do Carmo Machado, and E. S. De Almeida, “On the reliability of mapping studies in software engineering,” J. Syst. Softw., vol. 86, no. 10, pp. 2594–2610, Oct. 2013, doi: 10.1016/j.jss.2013.04.076.
17. H. Zhang, M. A. Babar, and P. Tell, “Identifying relevant studies in software engineering,” Inf. Softw. Technol., vol. 53, no. 6, pp. 625–637, 2011, doi: 10.1016/j.infsof.2010.12.010.
18. B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering,” 2007.
19. Z. Li, P. Avgeriou, and P. Liang, “A systematic mapping study on technical debt and its management,” J. Syst. Softw., vol. 101, pp. 193–220, Mar. 2015, doi: 10.1016/j.jss.2014.12.027.
20. C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE ’14, 2014, pp. 1–10, doi: 10.1145/2601248.2601268.
21. L. F. Restrepo Gutierrez, “Replication package for: ‘Snapshot’ of the State of Software Reuse in Colombia,” vol. 1, 2021, doi: 10.17632/2HDX42X6WC.1.
22. X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, “A map of threats to validity of systematic literature reviews in software engineering,” in Proceedings - Asia-Pacific Software Engineering Conference, APSEC, Jul. 2016, vol. 0, pp. 153–160, doi: 10.1109/APSEC.2016.031.
23. D. Lucrédio et al., “Software reuse: The Brazilian industry scenario,” J. Syst. Softw., vol. 81, no. 6, pp. 996–1013, Jun. 2008, doi: 10.1016/J.JSS.2007.08.036.
24. R. Capilla, B. Gallina, C. Cetina, and J. Favaro, “Opportunities for software reuse in an uncertain world: From past to emerging trends,” J. Softw. Evol. Process, vol. 31, no. 8, p. e2217, Aug. 2019, doi: 10.1002/SMR.2217.
25. D. Bombonatti, M. Goulão, and A. Moreira, “Synergies and tradeoffs in software reuse – a systematic mapping study,” in Software - Practice and Experience, Jul. 2017, vol. 47, no. 7, pp. 943–957, doi: 10.1002/spe.2416.

26. M. Morisio, M. Ezran, and C. Tully, “Success and failure factors in software reuse,” IEEE Trans. Softw. Eng., vol. 28, no. 4, pp. 340–357, Apr. 2002, doi: 10.1109/TSE.2002.995420.

27. D. C. Rine and R. M. Sonnemann, “Investments in reusable software. A study of software reuse investment success factors,” J. Syst. Softw., vol. 41, no. 1, pp. 17–32, Apr. 1998, doi: 10.1016/S0164-1212(97)10003-6.

28. V. Garcia et al., “Towards a Maturity Model for a Reuse Incremental Adoption,” SBCARS, Oct. 2007, doi: 10.6084/M9.FIGSHARE.96661.

29. W. Spoelstra, M. Iacob, and M. van Sinderen, “Software reuse in agile development organizations: a conceptual management tool,” in Proceedings of the 2011 ACM Symposium on Applied Computing - SAC ’11, 2011, p. 315, doi: 10.1145/1982185.1982255.

30. D. C. Rine and N. Nada, “Empirical study of a software reuse reference model,” Inf. Softw. Technol., vol. 42, no. 1, pp. 47–65, Jan. 2000, doi: 10.1016/S0950-5849(99)00055-5.

31. W. B. Frakes and C. J. Fox, “Sixteen questions about software reuse,” Commun. ACM, vol. 38, no. 6, p. 75-ff., Jun. 1995, doi: 10.1145/203241.203260.

32. M. A. A. Rothenberger, K. J. J. Dooley, U. R. R. Kulkarni, and N. Nada, “Strategies for software reuse: a principal component analysis of reuse practices,” IEEE Trans. Softw. Eng., vol. 29, no. 9, pp. 825–837, Sep. 2003, doi: 10.1109/TSE.2003.1232287.

33. J. K.S and V. R, “A New Capability Maturity Model For Reuse Based Software Development process,” IACSIT Int. J. Eng. Technol., vol. 2, no. 1, p. 5, 2010, Accessed: Jun. 08, 2018. [Online]. Available: http://www.ijetch.org/papers/108-T237.pdf.

34. M. Morisio, C. Tully, and M. Ezran, “Diversity in reuse processes,” IEEE Softw., vol. 17, no. 4, pp. 56–63, 2000, doi: 10.1109/52.854069.

35. L. Buglione, G. Lami, C. G. von Wangenheim, F. M. Caffery, and J. C. R. Hauck, “Leveraging Reuse-Related Maturity Issues for Achieving Higher Maturity and Capability Levels,” in Leveraging Reuse-Related Maturity Issues for Achieving Higher Maturity and Capability Levels, Springer, Berlin, Heidelberg, 2013, pp. 343–355.

36. L. Bass, C. Buhman, S. Comella-Dorda, F. Long, and J. Robert, “Volume 1: Market Assessment of Component-Based Software Engineering.” Software Engineering Institute, 2000, Accessed: Jun. 29, 2018. [Online]. Available: http://www.dtic.mil/docs/citations/ADA395250.

37. L. Rincon, R. Mazo, and C. Salinesi, “APPLIES: A framework for evaluAting organization’s motivation and preparation for adopting product lines,” in 2018 12th International Conference on Research Challenges in Information Science (RCIS), May 2018, pp. 1–12, doi: 10.1109/RCIS.2018.8406641.

38. C. Catal and Cagatay, “Barriers to the adoption of software product line engineering,” ACM SIGSOFT Softw. Eng. Notes, vol. 34, no. 6, p. 1, Dec. 2009, doi: 10.1145/1640162.1640164.

39. T. Käköla and J. C. Duenas, Eds., Software Product Lines. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

40. [C. Sholom G, “Product Line State of the Practice Report,” Software Engineering Institute, 2002. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5961 (accessed Jul. 23, 2018).

41. M. Jha and L. O’Brien, Identifying Issues and Concerns in Software Reuse in Software Product Lines, vol. 5791. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

42. C. Palomares, C. Quer, and X. Franch, “Requirements reuse and requirement patterns: a state of the practice survey,” Empir. Softw. Eng., vol. 22, no. 6, pp. 2719–2762, Dec. 2017, doi: 10.1007/s10664-016-9485-x.

43. H. Züllighoven, “12 - The Development Process,” in Object-Oriented Construction Handbook, H. Züllighoven, Ed. San Francisco: Morgan Kaufmann, 2005, pp. 393–457.

44. R. J. Leach, Software Reuse, Second Edition: Methods, Models, Costs, Second. New York, NY: McGraw-Hill New York, 2012.

45. D. Fafchamps, “Organizational Factors and Reuse,” IEEE Softw., vol. 11, no. 5, pp. 31–41, 1994, doi: 10.1109/52.311049.

46. W. C. Lim, “Legal and contractual issues in software reuse,” in Proceedings of Fourth IEEE International Conference on Software Reuse, 1996, pp. 156–164, doi: 10.1109/ICSR.1996.496123.

47. V. C. Garcia et al., “Towards an Assessment Method for Software Reuse Capability (Short Paper),” in 2008 The Eighth International Conference on Quality Software, Aug. 2008, pp. 294–299, doi: 10.1109/QSIC.2008.58.

48. F. van der Linden, K. Schmid, and E. Rommes, Software Product Lines in Action. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

49. K. Sherif and A. Vinze, “Barriers to adoption of software reuse: A qualitative study,” Inf. Manag., vol. 41, no. 2, pp. 159–175, Dec. 2003, doi: 10.1016/S0378-7206(03)00045-4.

50. R. Keswani, S. Joshi, and A. Jatain, “Software Reuse in Practice,” in 2014 Fourth International Conference on Advanced Computing & Communication Technologies, Feb. 2014, pp. 159–162, doi: 10.1109/ACCT.2014.57.