The Journal of Systems and Software 111 (2016) 128–148

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Twenty-eight years of component-based software engineering

Tassio Vale [a,b,c,*], Ivica Crnkovic [e], Eduardo Santana de Almeida [b,c], Paulo Anselmo da Mota Silveira Neto [c,d], Yguaratã Cerqueira Cavalcanti [c,d], Silvio Romero de Lemos Meira [d]

[a] Center of Exact Sciences and Technology, Federal University of Recôncavo da Bahia, Cruz das Almas, BA, Brazil
[b] Computer Science Department, Federal University of Bahia, Salvador, BA, Brazil
[c] RiSE, Reuse in Software Engineering, Recife, PE, Brazil
[d] Informatics Center, Federal University of Pernambuco, Recife, PE, Brazil
[e] School of Innovation, Design and Engineering (IDT), Mälardalen University (MDH), Västerås, Sweden

ARTICLE INFO

Article history:
Received 18 July 2014
Revised 20 July 2015
Accepted 14 September 2015
Available online 25 September 2015

Keywords:
Systematic mapping study
Component-based software engineering
Component-based software development
Software component

ABSTRACT

The idea of developing software components was envisioned more than forty years ago. In the past two decades, Component-Based Software Engineering (CBSE) has emerged as a distinguishable approach in software engineering, and it has attracted the attention of many researchers, which has led to many results being published in the research literature. There is a huge amount of knowledge encapsulated in conferences and journals targeting this area, but a systematic analysis of that knowledge is missing. For this reason, we aim to investigate the state-of-the-art of the CBSE area through a detailed literature review. To do this, 1231 studies dating from 1984 to 2012 were analyzed. Using the available evidence, this paper addresses five dimensions of CBSE: main objectives, research topics, application domains, research intensity and applied research methods. The main objectives found were to increase productivity, save costs and improve quality. The most addressed application domains are homogeneously divided between commercial-off-the-shelf (COTS), distributed and embedded systems. Intensity of research showed a considerable increase in the last fourteen years. In addition to the analysis, this paper also synthesizes the available evidence, identifies open issues and points out areas that call for further research.

1. Introduction

Component-Based Software Engineering (CBSE) promotes the development of software systems through construction from existing software components, the development of components as reusable entities, and system evolution realization by the customization and replacement of components (Szyperski, 2002). The CBSE idea is not new. It was envisioned more than forty years ago by McIlroy (1968) who provided an idea of commercial component production similar to that found in other engineering fields. However, most of the research work on CBSE has emerged in the last two decades. The motivations behind CBSE have been of a business and technical nature, i.e. increased efficiency and effectiveness, lower costs, and shorter time-to-market on one side, and improved quality with regards to fewer errors, improved performance, maintainability, portability, etc. on the other side. However, for many of these claims there is no proper evidence. Furthermore, after many years of development, the question arises whether the research in this field has fulfilled its goals, or if there are still important issues that require further research. Regarding these years of development in the CBSE research field, Schneider and Han (2004) argue that a literature review on CBSE is important in order to investigate the state-of-the-art, pointing out research topics which researchers and practitioners should investigate.

In this context, we present a systematic mapping study (Kitchenham and Charters, 2007; Petersen et al., 2008) performed to map out the CBSE area in the period 1984–2012. The goal is to synthesize evidence of contributions from academic publications. We identify the existing research trends, open issues, and areas for further improvement.

The remainder of this paper is organized as follows: Next, Section 2 describes the mapping process undertaken in this study. Sections 3 and 4 present data analysis and synthesis results. Section 5 considers the threats to the validity of this research, and Section 6 presents the related work. Finally, Section 7 presents a summary of the work and directions for future research.

* Corresponding author at: Centro de Ciências Exatas e Tecnológicas (CETEC) da Universidade Federal do Recôncavo da Bahia (UFRB) - Rua Rui Barbosa, 710, Centro, Crus das Almas/BA, Brasil, CEP 44.380-000 - Phone: +55 75 3621 9747.
E-mail address: tassio.vale@ufrb.edu.br, tassio.vale@gmail.com (T. Vale).

http://dx.doi.org/10.1016/j.jss.2015.09.019
0164-1212/

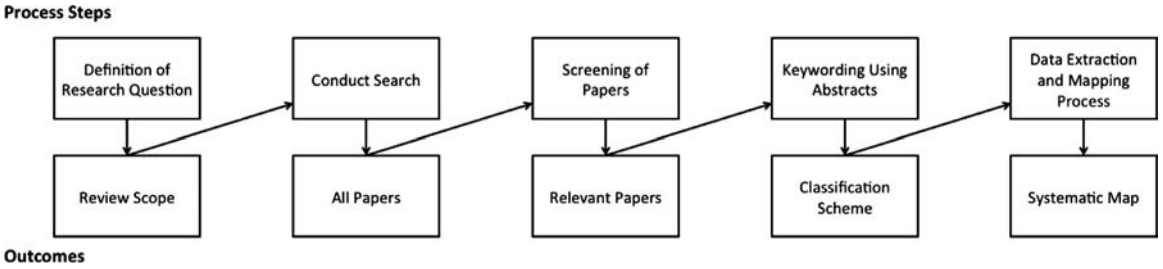

Fig. 1. Mapping study process (Petersen et al., 2008).

Table 1
List of journals and magazines with their h-indexes on 23 Dec 2013.

Acronym	Journal and magazines	h-index
CACM	Communications of the ACM	118
IEEE TSE	IEEE Transactions on Software Engineering	100
-	IEEE Software	67
JSS	Journal of Systems and Software	54
TOSEM	ACM Transactions on Software Engineering and Methodology	49
IST	Information and Software Technology	48
SPE	Software - Practice & Experience	45
STVR	Software Testing, Verification and Reliability	29
JSME	Journal of Software Maintenance and Evolution	29
SQJ	Software Quality Journal	21
-	IEEE Computer	Not found

2. Systematic mapping study method

The review method is based on guidelines for performing systematic mapping studies Petersen et al. (2008) and systematic literature reviews Kitchenham and Charters (2007). We adopted the mapping process defined in Petersen et al. (2008), as illustrated in Fig. 1.

The definition of research questions reflects the goal of this study. CBSE papers are identified by following search procedures, and the screening of these papers is the primary focus of this study. Furthermore, keywording using abstracts helps to develop the classification scheme and ensure that the scheme takes the existing studies into account. This classification will be used for data extraction from the primary studies, with the aim of synthesizing all data through the mapping process.

2.1. Research questions

The main question which drives this mapping study and reflects our goal is: "*How can we characterize the contributions of CBSE in the last twenty-eight years?*". In order to answer this question, five different questions were formulated, addressing different aspects of the CBSE area. The research questions are described as follows:

- **RQ1. What is the intensity of research activity on CBSE?**
  Rationale: In this question, "*intensity*" is represented by a quantification of studies, journals, conferences and countries that are involved in CBSE publications. This analysis indicates the intensity of research activities and identifies CBSE research communities.
- **RQ2. Which are the main objectives of applying CBSE?**
  Rationale: This question investigates the main technical and non-technical objectives addressed in the CBSE studies, such as productivity improvement, quality improvement or cost savings. The limitations of applying CBSE are also identified in this question.
- **RQ3. Which are the most frequently investigated CBSE research topics and how they have changed over time?**
  Rationale: This question analyzes research topics addressed by the CBSE studies. The research topics identified are further investigated and used for identifying research gaps.
- **RQ4. In which domains has CBSE been applied?**
  Rationale: This question aims at finding application domains in which CBSE has been most frequently applied, i.e., the specifics of CBSE utilization.
- **RQ5. Which are the most frequently applied research types and methods? How have they changed over time?**
  Rationale: This question identifies the applied research approaches and empirical research methods, analyzing how they have evolved over the years.

2.2. Conduct search

Although Kitchenham and Charters (2007) and Petersen et al. (2008) advocate the use of two search strategies, manual and automatic, manual search was considered as the primary source, given the infeasibility of analyzing all studies collected from automatic search. Therefore, we decided to perform a manual search which considers relevant journals and conferences related to the CBSE area.

The use of manual search as a unique way to search primary studies is commonly used in the literature (Jorgensen and Shepperd, 2007; Sjoberg et al., 2005). In fact, according to Brereton et al. (2007), current academic search engines are not designed to support systematic literature reviews.

Tables 1 and 2 show the journals and magazines with their h-indices[1] and conferences/workshops[2] that have been used in the study. These journals, conferences and workshops make up the vast majority of the publications related to CBSE [3].

2.3. Selection criteria

Each study was analyzed according to the defined inclusion and exclusion criteria, in order to verify whether it was suitable to address the research questions. The criteria are described as follows:

[1] H-index reference journals - http://www.scimagojr.com/
[2] H-index reference for conferences - http://shine.icomp.ufam.edu.br/
[3] By using keywords "software" and "component" and searching for studies in SCOPUS database, the first 200 most-cited relevant publications originate from the selected sources.

Table 2
List of conferences and workshops and their h-indexes on Dec 23 2013.

Acronym	Conference	h-index
ICSE	International Conference on Software Engineering	110
OOPSLA	Intl. Conf. on Object-Oriented Programming, Systems, Languages, and Applications	74
ASE	IEEE/ACM International Conference on Automated Software Engineering	52
ESEC/FSE	European Software Engineering Conference	48
SPLC	Software Product Line Conference	40
GPCE	International Conference on Generative Programming and Component Engineering	37
WICSA	Working IEEE/IFIP Conference on Software Architecture	32
SEAA	Euromicro Conference on Software Engineering and Advanced Applications	31
CBSE	International ACM SIGSOFT Symposium on Component Based Software Engineering	25
TOOLS	Technology of Object-Oriented Languages and Systems	21
ICCBSS (CCBS)	IEEE International Conference on COTS-Based Software Systems	20
ICSR	International Conference on Software Reuse	20
ECSA	European Conference on Software Architecture	15
SAVCBS	Specification and Verification of Component-Based Systems Workshop	7
QoSA	Conference on the Quality of Software Architectures	2
WCOP	Workshop on Component-Oriented Programming	Not found

- **Inclusion Criteria**. All studies published in the selected journals and conferences which address any CBSE topic; primary studies based on the same data and different focus are considered as different studies.
- **Exclusion Criteria**. gray literature (e.g. books, technical reports, etc.); short papers (fewer than five pages); studies which address only commercial-off-the-shelf (COTS) products without mentioning software components; studies which address components only as architectural units[4]; when the same study is reported by more than one paper, the most complete study was included.

2.4. Screening of papers

The selection of studies comprises a screening process composed of two iterations (see Fig. 3). These iterations were performed using peer review. Peer review involved two researchers assessing the same paper, and if there was any conflict of judgment, a third researcher resolved it. The relevant studies selected from this screening process are called primary studies (Kitchenham and Charters, 2007).

After performing manual search on the selected conferences and journals, 1396 studies were identified. The inclusion and exclusion criteria were applied in the first iteration on the title and abstracts. In the second iteration, a more detailed review was performed, by reading the introduction and conclusion of the papers, resulting in 1231 primary studies.

2.5. Classification scheme

Instead of using keywording with abstracts (Petersen et al., 2008), we adopted two taxonomies from other studies. They are described as follows:

- **Research Topic.** Schneider and Han (2004) summarize six research topics in the CBSE field based on Nierstrasz and Meijler (1995): *functionality, interaction, quality, management, evolution and tools* and *methodology*. (each category is described in Table 4). This is used to answer the RQ2 research question.
- **Research Type.** To analyze which type of research is provided in CBSE research literature, we adopt an approach from Wieringa et al. (2005), originally applied on requirements engineering. In Wieringa et al. (2005) a software engineering lifecycle is discussed, and related research activities are identified: (i) problem investigation (i.e. which problems exist in CBSE), (ii) solution validation (i.e. what are the properties of a proposed solution?), and (iii) implementation evaluation (i.e. what are the experiences with this implemented solution?). According to these activities, in Wieringa et al. (2005) the following classification of the research type has been derived: *evaluation, validation, solution proposal, philosophical argumentation, opinion* and *experience report*. See Table 5 for more details. This classification is used to answer the RQ5 research question.
- **Contribution Type.** The contribution types are based on Petersen et al. (2008). They are *method, process, modeling, technology* and *metric*. However, they were complemented with a new facet, *discussion*, since some of the primary studies did not fit Petersen et al. (2008) classification. See Table 6 for more details.

2.6. Data extraction

We designed a data extraction form[5] to collect all the information required to address the research questions and study objectives. The data extracted are: *paper ID, paper title, authors*, venue (*conference* or *journal*), *publication year, involved countries, research topic, specific topics, main objectives of applying CBSE, domain, research type, research method* and *contribution type*.

3. Data analysis and synthesis

In this section, we present the answers to the research questions by analyzing and synthesizing the data extraction results. In order to better understand the results, we grouped them according to each research question. Given space constraints, the primary studies cited in this section are represented by an ID (e.g. P145, P233), and further information about them is on the web[6]. A list of the the cited primary studies in available in Appendix A.

3.1. RQ1. What is the intensity of research activity on CBSE?

In the first question, four aspects were investigated: the evolution of publications over the years, the conferences and journals addressed, the countries involved, the citations, and the most influential authors (in publishing and citations).

A total of 1231 publications are spread across the years 1984 and 2012. Fig. 2 shows the distribution of these studies over the 28-year period. CBSE exhibited a linear growth until 2006, when 115 studies

[4] Components are intrinsic parts of software and there are many publications referring to components and software architecture. However, in CBSE, components have extended meaning and purpose when compared to "just" being part of an architecture.

[5] The data extraction spreadsheet is available at http://tassiovale.com/JSS-ValeEtAl-DataExtraction.html (File size: 2.4MB)

[6] Available at http://tassiovale.com/JSS-ValeEtAl-DataExtraction.html (File size: 2.4MB)

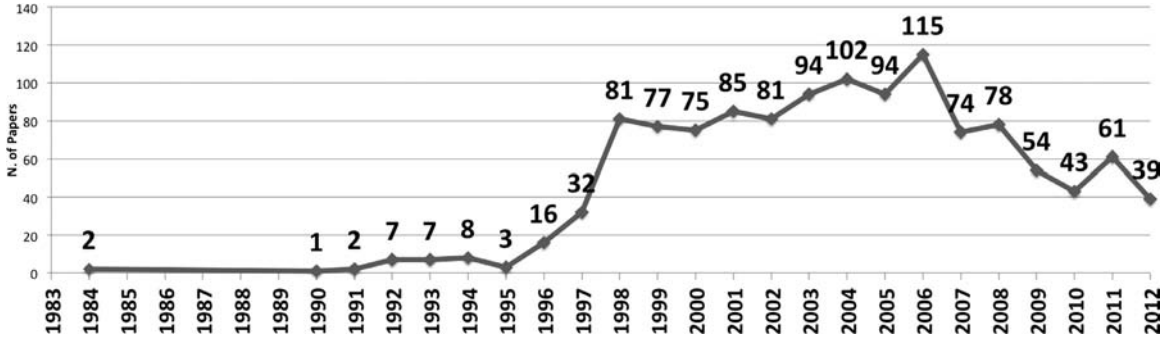

Fig. 2. Number of publications per year.

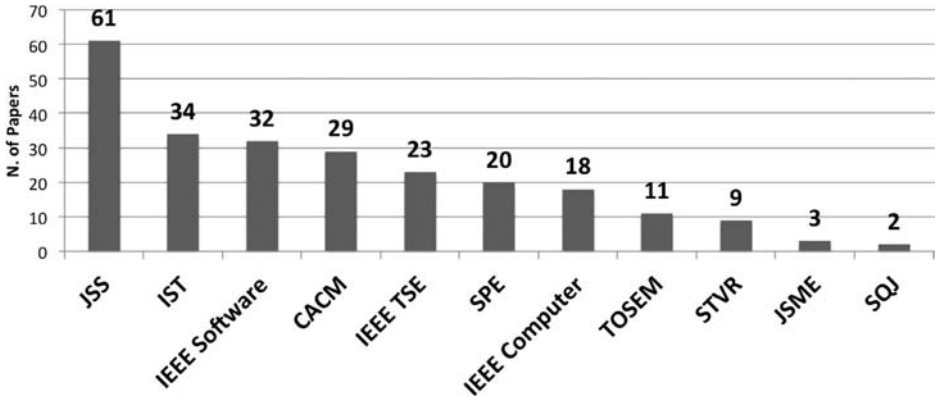

Fig. 3. Addressed journals.

were published. Intensity of research activity on CBSE was moderate until 1997, showing considerable growth in 1998, when the number of primary studies more than doubled. After 2006, there was a reduction in CBSE studies.

This mapping covered 13 conferences, 11 journals, almost 50 countries and more than 1400 authors involved in CBSE research. By analyzing the number of primary studies per journal and conference (Tables 3 and 4), most of the studies published are conference papers (988, i.e. 80%) against 243 (20%) journal papers.

Regarding the journals and conferences which publish CBSE studies (see Figs. 3 and 4), the International Symposium on Component-Based Software Engineering (CBSE) and Journal of Systems and Software (JSS) were the top contributors. Both of them had almost twice the number of primary studies compared to those in second position.

CBSE is a research area investigated in most continents (see Fig. 5): Asia, Africa, North America, South America, Europe, and Oceania. Forty-nine different countries have published CBSE primary studies in different journals and conferences. Among them, the USA is the country which has the highest number of publications, 358 (29% of the total). It is followed by Germany, France and the United Kingdom, with 130, 117 and 97 publications respectively.

What is the impact of the CBSE papers? Fig. 6 shows the number of citation for the 100 most cited papers[7], and Table A.1 presents the complete list. There are ten papers that stand out with more than 450 citations each. The h-index of these papers is 97, i.e., there are 97 studies cited at least 97 times. The total number of citations is 40,1330, and the average number of citations is 33. To these citations we should add three books not included in the list of selected studies (due to being excluded by the exclusion criteria), but which had a great impact on CBSE research and practice: Szyperski (2002) with 7450 citations, Heineman and Councill (2001) with 1213 citations, and Crnkovic and Larsson (2002) with 591 citations.

Table 7 shows the top 47 authors' citations for the selected primary studies.

Summary: RQ1. What is the intensity of research activity on CBSE?

The distributions clearly show an increasing interest in CBSE in the late nineties, keeping the interest high in 2000s, and a lower, but still stable interest in the 2010s. *Observation:* There are several reasons for the lower publication rates: the process for reviewing and publishing CBSE primary studies is more rigorous, since there is already a large body of knowledge; CBSE has been integrated into other approaches, such as service-oriented development, software product lines and model-based engineering; and new approaches (service-oriented development, cloud).

3.2. RQ2. Which are the main objectives of applying CBSE?

The first question aims to investigate the technical and non-technical objectives of CBSE. By technical objective we mean the concerns and results based on technical challenges and solutions related to engineering (e.g. specification and implementation of interfaces and components interactions through the interface), while non-technical objectives include concerns related to other areas (e.g. business, organization, ability, etc.). Only 33% of the studies (411 primary studies) explicitly reported the main objectives that are often related to benefits that CBSE can offer.

Fig. 7 shows the most commonly referred objectives: increasing productivity, cost savings, increasing quality, increasing reusability, decreasing complexity, increasing maintainability, increasing flexibility, decreasing risks, increasing efficiency, increasing adaptability and

[7] Source: Google Scholar, 10 June 2014

Table 3
Screening process.

Iteration	Action	No. of papers
Iteration 1	Include studies on the basis if the inclusion criteria applied to titles and abstracts	N = 1396
Iteration 2	Exclude studies by applying then exclusion criteria on introductions and conclusions	N = 1231

Table 4
Research topics (Schneider and Han, 2004).

Research Topics	Description
Functionality:	This category represents proposals to characterize software components; abstractions, languages and notations to specify functional properties of components. They can be general-purpose or domain-specific components.
Interaction:	Suitable ways of expressing compositions of components are presented; languages for composition; implicit/explicit, declarative/imperative composition; it investigates how composition helps in application evolution; correctness of component compositions given a set of requirements; interaction protocols.
Quality:	Specification of (non-functional) quality attributes of components and verification of these attributes given their implementations; quality attributes of a composition of components given the quality attributes of the components involved (i.e. can we perform compositional reasoning).
Management:	Ontologies to create suitable component repositories; how effectively to search for components given a set of requirements.
Evolution and Tools:	Software tools, development environments etc. to facilitate Component-Based Software Engineering; how to manage framework development and component repositories to provide the best possible support for both component engineers and application developers.
Methodology:	Primary studies in this category investigate the impact of component- and reuse-based development on software engineering methodologies; development processes and methods to facilitate component-based software engineering; the impacts of CBSE from a business perspective.

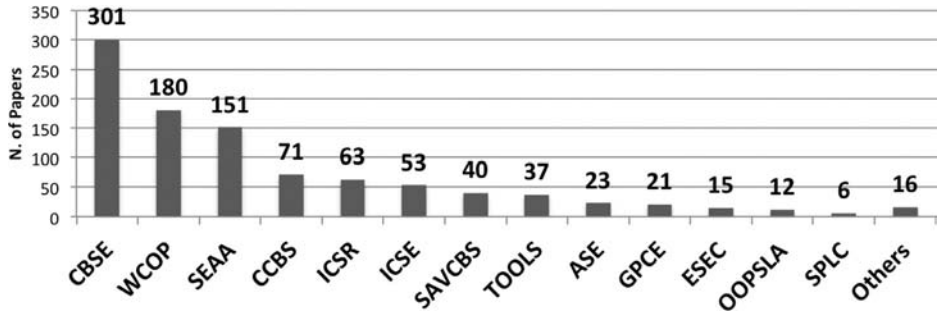

Fig. 4. Addressed conferences.

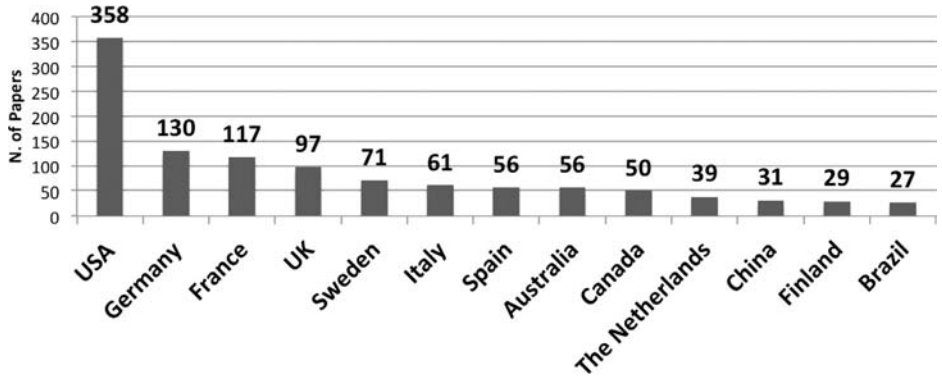

Fig. 5. Top countries.

optimized evolution. The majority of objectives are of non-functional character, including non-technical (productivity and cost as business issues), and the others being technical issues related to software quality. We provide short essence of the objectives extracted from the primary studies as follows.

Increasing productivity. This objective is related to shorten system development time and consequently, reduced time-to-market. Systems are no longer built from scratch, and developers can save time using prefabricated components, through a larger-grained software reuse (P529; P443).

Using pre-existing, well designed, tested, and documented elements as components of programs amplifies the programming capabilities, reduces the amount of work needed on new programs and systems, and achieves better overall control over the production process and the quality of its products (P412; P452). Most of the primary studies in their motivations and analysis refer to system development with reuse, while there are a few studies that measure the effort involved in building for reuse (i.e. components), or total effort. The study (P496) refers to industry experience that building reusable components on average requires three to five times more effort than building the same function but not building for reuse. This is one of the main challenges in starting to apply CBSE (P890).

Cost savings. Most studies mention productivity increasing and costs reduction together as main objectives. CBSE can be used

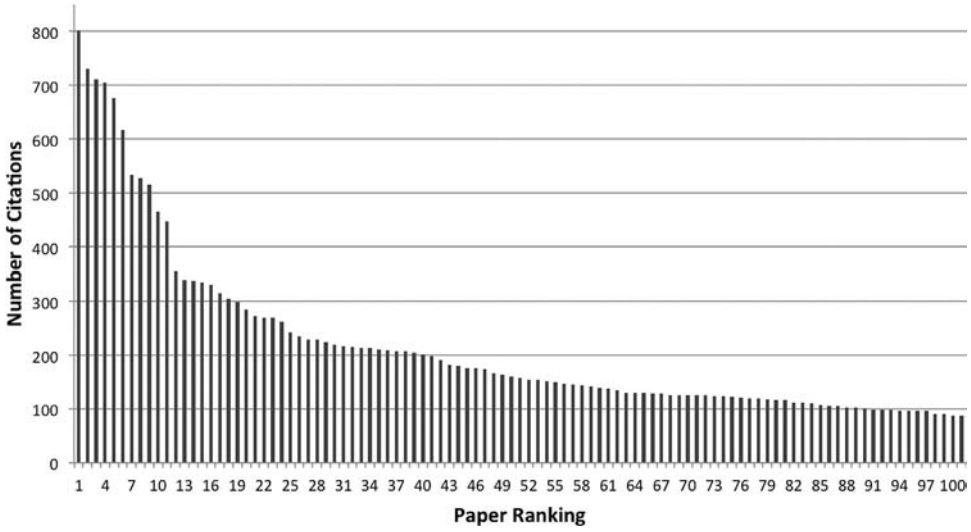

Fig. 6. Number of citations for the 100 most cited papers.

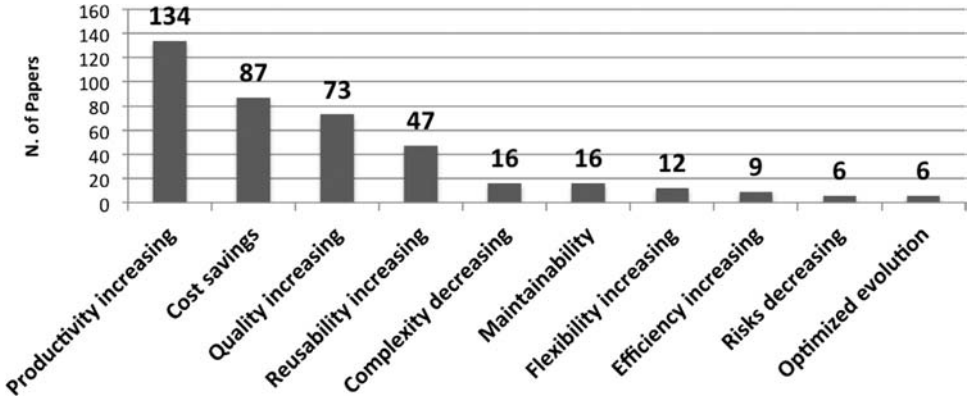

Fig. 7. Main objectives of applying CBSE.

potentially to reduce costs, as well as reducing software development time by bringing the system to market as early as possible (P457).

Such savings are related to software development and maintenance. Rather than starting from scratch with new development efforts, an emphasis must be placed on using and integrating pre-fabricated software components. This approach avoids the duplication of work and lowers the overall development costs associated with the construction of new software applications (P413; P630). On the other side, maintenance of reusable components may lead to increased costs (P1216), which can be paid off by higher reuse.

Voas (P6) argues "*the savings for developers could be staggering. If a world-class programmer costs $500 per day, purchasing 100,000 lines of code would result in saving $5 million (minus licensing fees). Companies producing components would also see great rewards. If $1 million were an acceptable licensing fee for 100,000 lines of code, then only five licenses would pay for that component, and the producer would profit on subsequent sales.*". This analysis is in line with the experience in industry (P496).

Increasing quality. This objective can be achieved by reusing of software components already tested and validated (P514).

Increasing reusability, increasing flexibility and optimized evolution. Wang et al. (P14) address three main objectives: increasing reusability, increasing flexibility and optimized evolution. According to him, the separation of concerns in CBSE substantially increases component reusability and provides more flexibility in system design and evolution.

Decreasing complexity. The means for decreasing complexity in CBSE is reusing components with encapsulated functions, preferably treated as black boxes, and providing standardised rules for component compositions through component interfaces. (P36,P365).

Increasing maintainability. Maintainability is improved in CBSE by providing easy replacement of obsolete components with new enhanced ones (P444). Maintainability of components also improves, but only after a period during which the component reaches "maturity". During the initial period of reuse a component may require more adoption to different products before its use becomes sufficiently widespread, at which point it enters a more stable phase (P1024).

Increasing efficiency. It is more efficient composing software systems by using components rather than building systems with non-modular solutions (P518).

Increasing adaptability. The means for increased adaptability are typically achieved by component adaptation techniques such as using adapters and wrappers (P319), or by building middleware that adapt to different protocols (P603; P670; P786)

Decreasing risks. Some studies (for example P7; P181), claim that by reusing components and reusing architectural solutions that component models provide, in general the risks (for lower quality, for increased costs, or time-to-market problems) decrease. However no empirical evidence is provided in these studies.

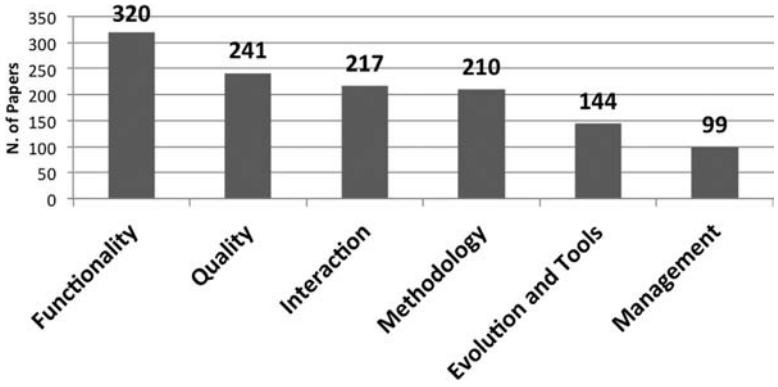

Fig. 8. Addressed research topics.

In addition to the objectives related to improvements in the development process, product quality or better development performance, in a smaller number of studies the inherited limitations and disadvantages are discussed. The characteristics of these limitations are described in Crnkovic (2001). In addition, several studies address some of these limitations and problems in their research, which are specified in the following:

- *Traceability.* Generality and interoperability of components and their black box characteristics create problems, such as architectural mismatches, and hidden behavior not visible from component specifications. When such problems appear in component-based systems, they are very difficult to trace (P77). Traceability in this context means the ability to identify problems across components given their black-box nature.
- *Compatibility.* Difficulty in finding compatible software components, either for the previous versions, or for the development or run-time environment (P201).
- *High initial efforts.* The development and the run-time environment for efficient use of components is complex and requires additional efforts and skills (P34).
- *Information problem.* Insufficient information exchange between a component developer and component consumer. In specification languages used in many component models the component specification is not sufficient for understanding its behavior (P307,P653).
- *Long-term component management.* Many important issues related to long-term system evolution and its support remain unsolved or unclear in relation to the components used in the system (component evolution support responsibility, ability to evolve with the system evolution, etc.) (P890).

Summary - RQ2. Which are the main objectives of applying CBSE?

The main objectives, explicitly addressed, of applying CBSE are of a non-technical nature. The main driving forces for CBSE are time-to-market, and increased productivity during system development. The quality issues related to both technical and non-technical solutions are also in focus as objectives of CBSE. The means of achieving these objectives are, however, in the vast majority of cases of a technical nature (e.g. introduction of a new technology, or a specification language, dynamic uploading, etc.). *Observation:* In most of the studies the main CBSE objectives are not referred to, but, rather, are implicitly assumed. Very few studies provide empirical evidence of the objective achievements.

3.3. RQ3. Which are the most investigated CBSE research topics and how they have changed over time?

This question identified the most addressed CBSE topics and their evolution over the years. To answer to this question we adopt a classification from Schneider and Han (2004), who identified the six main research topics in CBSE:

- *Functionality and specification.* This concerns the components – What are the characteristics of the components? What kind of abstractions, languages, etc. are used for specification of functional properties?
- *Interaction.* This focuses on component composites – What are suitable ways of expressing compositions of components, ensuring their correctness?
- *Quality.* How to specify quality attributes of components and verify that these attributes are met by a given implementation?
- *Management.* How to facilitate the design and implementation of component frameworks? How to store and search for components?
- *Evolution and tools.* What kind of software tools, development environments etc. are needed to facilitate CBSE?
- *Methodology.* What is the impact of the component-based development on software engineering methods?

Functionality was the most frequently covered topic in 320 primary studies, followed by *quality* (241 primary studies), *interaction* (217 primary studies), *methodology* (210 primary studies), and *evolution and tools* (144 primary studies). *Management* was the least addressed topic, with 99 primary studies. Fig. 8 presents a summary of this distribution.

Regarding the evolution of topics over the years, during the early period (1984–1997) there was not a significant difference in the number of primary studies per topic. Moreover, the rate of primary studies per topic began changing from 1998, when the research activity in the CBSE field started to grow.

Fig. 9 shows the evolution of each research topic over the years. In order to obtain more precise evidence, a set of specific research topics is also identified. Most of the primary studies were classified in just one specific topic, and the remaining ones were classified in two different topics.

Using the keyword technique (Petersen et al., 2008), the specific topics were identified. Table 8 shows them grouped in the topics classified according to Schneider and Han (2004) (shown in Table 4).

Fig. 10 shows the eight most frequently addressed specific topics: component composition, component models, software architecture, frameworks, tools, component search and retrieval, testing and quality. Primary studies explaining each specific topic were selected based on the strong relationship between their objectives and the specific topic in question.

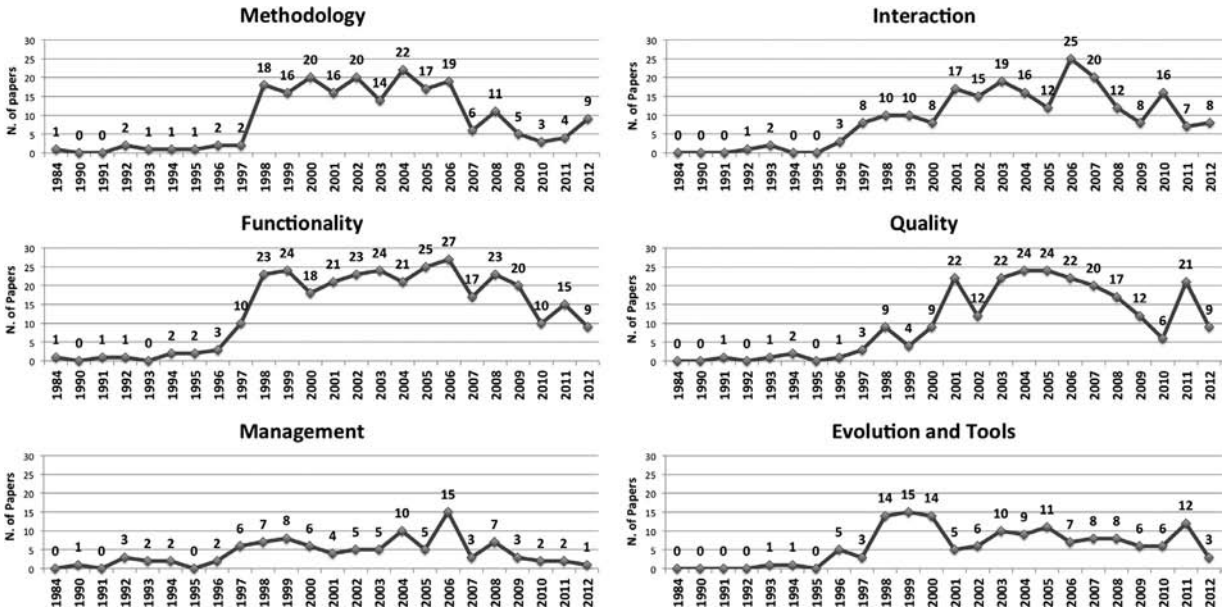

Fig. 9. Research topic by year.

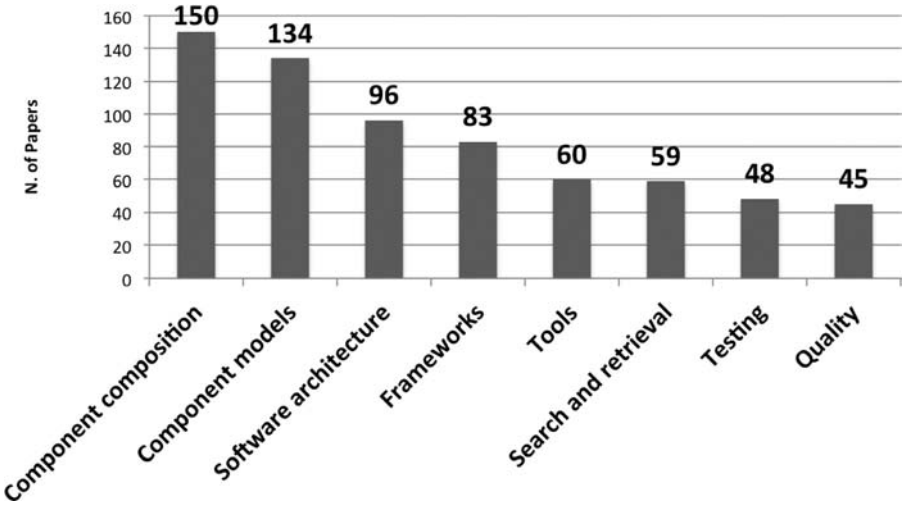

Fig. 10. Top 8 specific topics.

Below we discuss the research topics addressed the most frequently in the primary studies.

3.3.1. Interaction: component composition

Component composition is the process of constructing applications by interconnecting software components through their interfaces (Nierstrasz and Meijler, 1995). Interaction (P641), assembly (P14; P588), integration (P303; P363; P688), binding (P1107) and composition (P49; P369; P585; P982) are all terms found in the primary studies. They correspond to the same concept "composition" since the studies are addressing issues related to the interconnection of components.

Furthermore, four studies propose techniques to ensure the quality of compositions. These techniques are intended to validate (P77), evaluate (P562), ensure integrity (P33) and detect vulnerabilities (P654) of connections among software components.

A set of primary studies investigate and improve elements responsible for connecting software components: interfaces (P150; P355; P453; P594; P693), connectors (P350; P540) and adapters (P203; P469).

Software components are composed several times during a system lifecycle. In order to meet new business requirements, components are modified and must replace older ones. Substitution mechanisms are applied to solve this problem (P21; P304; P1105). This substitution can be performed in a static way, where components are composed before running, or dynamically, where components are updated during execution (P67; P618; P1036; P1101).

In addition to the composition of functional properties (i.e. binding, or functional composition), many studies discuss the composition of particular non-functional properties, for example real-time properties, resource usage, performance (P170; P187; P495; P395; P441), or the composition of non-functional properties in general (P398; P809).

3.3.2. Functionality: component models

A component model defines a set of standards for component implementation, naming, interoperability, customization, composition, evolution, and deployment (Heineman and Councill, 2001). There are plenty of component models (i.e. more than 50) in the literature referred as state of the practice and state-of-the-art.

Lau and Wang (2005/2006, P638; P341) propose a taxonomy of component models that enables characterization of component models in relation to their specification, interaction and composability. Component models are divided into (i) component models in which components are classes, (ii) models in which components are objects, and (iii) those in which components are architectural units. Crnkovic et al. (2011, P653) present a framework for classifying component models, where different dimensions of component model characteristics are identified: (i) a lifecycle dimension (characterized by architectural modeling & design, implementation, storage, deployment), (ii) interconnection (whose characteristics are interface, binding, and interaction), and (iii) extra-function properties (characterized by management, specification and composability). In addition, a classification for twenty different component models is presented. Brada (2009, P390) reports a survey of typical representatives of industrial and research component models.

We identified several component models, and the most frequently addressed are: the Fractal, "Lau et al." component model, the ProCom and the CORBA Component Model (CCM).

The Fractal is a general component model which is intended to implement, deploy and manage complex systems, in particular operating systems and middleware (P979). It is a hierarchical and reflective model with sharing, supporting a set of component control capabilities (P345; P688; P979; P1038; P1049; P1114).

In the Lau et al. (2006) component model, components encapsulate computation (and data), and composition operators encapsulate control. Therefore, components are constructed from computation units and connectors. Moreover, there are proposals for improvement of this model (P355; P658; P1063; P1064; P1102; P1113; P1129).

ProCom is a two-layer component model for design and development of embedded systems with the aim of using CBSE for decreasing the complexity in design and providing the grounds for analyzing them and predicting their properties, such as resource consumption and timing behavior (P1087). Other primary studies reported improvements for this component model (P657; P671; P668; P1130).

The CORBA Component Model (CCM) extends the CORBA object model and defines features that allow software developers to implement, manage, and deploy software units (components) that integrate commonly used functionality and allow for greater software reuse (P656). Extensions for this model were identified in this mapping study (P284; P285; P656; P661).

Several other component models were addressed: Microsoft COM (P425; P630), Koala (P16,P51), CoOWA (P604), KOM (P609), SC (P462), JavaBeans and OSGi (P286), Robocop (P311), SaveCCM (P333; P398), CSCM (P314), UM-RTCOM (P531), PECOS (P598), CAmkES (P525), MidGate (P366), PCM (P535; P845; P1221), SCA (P1036), RCM (P664), and others (P32; P61; P85; P122; P124; P141; P239; P310; P312; P536; P594; P598; P615; P667; P631; P1076).

3.3.3. Functionality: software architecture

Software architecture is closely related to components and component models, and software architecture issues have been addressed together with them. In addition particular software architectures were identified in order to structure the development and maintenance of component-based systems. They are: CASiNO (P27), IRL (P31), Boca (P549), Malaca (P483) and others (P113; P146; P247; P294; P340; P368; P465; P495; P551; P570; P571; P579; P977; P1015; P1058).

Moreover, there are proposals for domain-specific architectures. The addressed domains are: product lines (P54), avionics (P55) and COTS products (P147; P155; P180).

Aspects considered by component-based architectures are standardization and quality properties. Standards for these architectures are realized by patterns (P34), styles (P205), reference architectures (P412), architectural description languages (P254; P261; P632) or architectural description methods (P125; P324). Exploration and evaluation of quality properties (or non-functional properties) are addressed in CBSE (P288; P1030). Some of the primary studies also explore reusability at the architectural level, proposing the specification of component architectures by reusing available components, component interfaces and interconnections between components (P74; P105; P454).

3.3.4. Functionality: frameworks

The definitions of frameworks vary (P423). We adopted the Schmidt (1996, P542) definition: "*an application framework provides the basic functionality of a working application, which can be easily tailored to a specialized application*". The frameworks identified in this study are heterogeneous, and address different issues.

Certain groups can be highlighted: domain-specific frameworks, CORBA-based frameworks (P25; P345; P611), conceptual/semantic frameworks (P41; P316; P988), and frameworks that support runtime composition (P201; P539; P980).

The domain-specific frameworks aim at customizing the CBSE for specific domains. Such domains are: embedded systems (P351; P374), medical imaging (P52; P213), radio-astronomical imaging (P38), telecontrol applications (P37), and voice applications (P1019).

Besides these groups, there are several different proposals for frameworks in the primary studies (P85; P173; P154; P232; P299; P306; P329; P335; P386; P389; P410; P424; P427; P473; P542; P532; P582; P593; P602; P617; P651; P652; P1025; P1049; P1131). Two studies perform an analysis of component-based frameworks. Casagni and Lyell (2003, P233) compare and contrast two frameworks, the Java Enterprise Edition (J2EE) framework and the FIPA-compliant multi-agent system (MAS). Hosek et al. (2010, P1110), meanwhile, discuss the distinguishing features of selected modern component-based frameworks for real-time embedded systems. They identify important features for building systems from components in this domain.

3.3.5. Evolution and tools

CBSE's main concerns relate to technologies and tools, and tool support is inevitable for the success of CBSE. For this reason many studies emphasize the tool support. Such tools can be integrated development environments and platforms such as PCLAgenda (P591), Cadena (P231), DesCOTS (P318), Concerto platform (P320), PRIDE (P1219), Save-IDE (P1218), REMES (P681), or platforms such as CIF (P1118), and middleware (P656). The remaining tools for CBSE development are: PEEL (P207), GOPCSD (P101), SYNTHESIS (P245), Genie (P249), eC3M (P675), visualization tools (P326; P347; P361), Computer-Aided Software Engineering (CASE) tools (P397), architecture (P600), testing (P633), model-driven development (P284) and others (P29; P141; P396; P422; P608; P678).

Furthermore, Barn and Brown (1998, P597) examine the role of appropriate new methods and tools in encouraging the wider practice of CBSE. They also describe the development of a new generation of tools for CBSE.

3.3.6. Methodology: search, selection and retrieval

Since in CBSE the component development process is separated from the system development process, activities related to searching for and retrieving components, as well as preparations to facilitate these activities, are important parts in the entire CBSE process.

First, primary studies present proposals for the characterization of components before search, selection and retrieval. This involves identification (P186; P187; P276), ranking (P121; P229; P276), selection criteria (P564) and classification of components within a repository (P186; P187; P270; P628).

Second, they propose techniques for search and selection (P88; P160; P202; P485; P581; P592), specially the selection of COTS

components (P102; P133; P159; P170; P318). In addition, there are proposals for component retrieval (P86; P98; P116; P204; P416; P458; P512; P634). Nevertheless, two studies report problems when performing component selection (P198; P362). Consequently, a set of search engines (P189; P392; P1029; P1033) and systems (P99; P413; P506; P511) automatize techniques for component search, selection and retrieval. Some of the primary studies address a trade-off analysis in combination with component retrieval which component version is better suited to a particular system, or form a decision about whether it is better to develop or to buy a component (P1217).

In order to analyze the existing component search and retrieval techniques, Lucredio et al. (2004, P321) present a survey about the main research on component search. In addition, Mili et al. (2003, P466) perform an experiment in which different component retrieval methods are tested.

3.3.7. Quality: testing

Developers (re-)using a component need to do considerable testing to ensure the software behaves properly in its new environment. They should test the components both in isolation and once integrated (P548). Third-party testing should also be considered (P554).

Component testing proposals identified in this work comprise the following topics: interaction and integration testing, frameworks, black-box testing, COTS components testing, testing architecture (P40), test-driven development (TDD) (P1099), runtime testing (P144; P396), tools (P397; P633), discussion papers (P548; P554) and others (P44; P45; P142; P156; P200; P305; P388; P608; P625; P1039; P1138).

In terms of interaction and integration, Jaffar-ur Rehman et al. (2007, P48) propose the testing of a component when put in the context of other interacting components. Sundmark et al. (2008, P1090) describe the added test effort required by component integration, by introducing the concept of compositionally introduced test items. In addition, two frameworks are proposed, in order to test the component in a deployment environment (P232; P240).

Considering black-box approaches for component testing, Edwards (2001, P43) outlines a strategy for automated black-box testing of software components that includes automatic generation of component test drivers, blackbox test data, and automatic or semi-automatic generation of component wrappers. Zhang et al. (2006, P191) present a process for selection of regression tests for user/glue code that uses COTS components. Testing COTS components is addressed by two other studies (P244; P307).

3.3.8. Quality

Quality issues treated in CBSE have two aspects: either the absence of failures and related methods to achieve that goal as much as possible, or non- functional properties of components or component-based systems, with their specifications, modeling, compositions or measurements. The identified topics of component quality are: failure handling, quality of service (QoS), quality properties, trusted components (P143; P150; P608), correctness verification (P64; P986) and others (P327; P343; P372; P481).

Failure handling is performed by predicting (P241; P489), preventing (P250; P394), detecting (P328), monitoring (P507) or analyzing (P503) failures in component-based systems.

Furthermore, a set of studies develops solutions to satisfy QoS requirements of component-based systems (P516; P983; P1024; P1041). Quality properties (P352; P524; P1128), attributes (P1051) and criteria (P383) are used to evaluate software components, especially the reliability property (P222; P242; P248).

Mahmood et al. (2005, P475) present a survey of component-based system quality assurance and assessment. They analyze formalism, cost estimation, and assessment and measurement techniques for the following quality attributes: performance, reliability, maintainability and testability.

Summary - RQ3. Which are the most investigated CBSE research topics and how they have changed over time?

CBSE can be seen as an overall software engineering area – covering practically all concerns that are covered by SE (e.g. requirements engineering, architecting, testing, etc. for components and component-based systems). In addition there are concerns that are specific to CBSE, and these concerns are covered by more than 50% of CBSE-related studies. The dominant research topics are related to the specification (and understanding of principles) of components, component-based systems, component models, and interfaces, and to the question of component interactions and compositions. Interest for the most of the research topics remained on the same level through the entire period (with a significant jump of research contributions in late nineties) but new domains have been added. Methodology is an exception to that rule - it shows more results in early 2000 while less in 2010s. *Observation:* The studies show a vast diversity of concepts, theories, and implementations. This suggests that CBSE is still an active area for research with, as yet, not all goals achieved and established in practice.

3.4. RQ4. In which domains has CBSE been applied?

This question analyzed application domains addressed by CBSE. The keywording technique (Petersen et al. (2008)) was used to extract data for this question. Most primary studies do not mention/evaluate their application domains. There were 394 studies (32%) that explicitly addressed a particular domain, and in several cases they referred to multiple domains.

The software engineering domains identified are: commercial off-the-shelf (COTS) systems, embedded systems, real-time systems, distributed systems, web-based systems, software product lines (SPL), operating systems, aspect-oriented software development (AOSD), model-driven development (MDD), open-source software (OSS), service-oriented systems, programming languages, graphical user interface, legacy systems, safety-critical systems, mission-critical applications, large-scale systems, feature-oriented software development, voice applications, etc. Fig. 11 shows the ten most addressed domains, explained below.

When analyzing the business domains, the following areas have been addressed in the studies: telecommunication, automotive systems, industrial automation, avionics, healthcare systems, financial systems, space-based systems, consumer electronics software, educational systems, enterprise systems, mass-market applications, nuclear systems, process control systems, compiler systems, radio astronomical systems, games, multimedia systems, network applications, robotics, ubiquitous systems, government systems, cloud computing, etc.

COTS and OSS. COTS is the most frequently addressed domain. 100 studies apply CBSE to develop COTS systems. Reusing components from third- party providers is one key technology used to develop systems quickly and cost-efficiently. These components, also known as OTS (off-the-shelf) components, come in two different types: COTS and OSS components (P103).

CBSE has been adopted as one of the main approaches to develop COTS components. According to Morisio and Torchiano (2002), *"COTS products and components are two sets with a non-empty intersection"*. Primary studies regarding the intersection between COTS and software components are handled in several topics such as security (P8; P134; P153), integration (P9; P149; P158; P218), architecture (P135), methodologies (P152), component selection (P170; P171; P206; P553), testing (P240; P244), and so on.

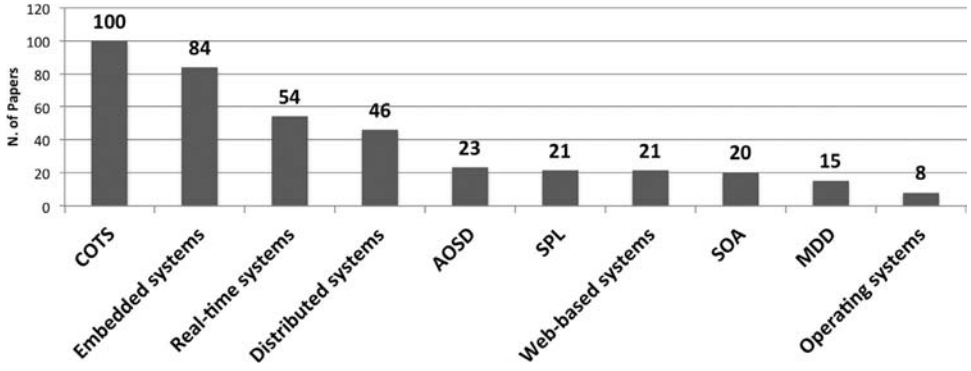

Fig. 11. Top 10 CBSE application domains.

On top of that, eight studies address OSS components. In many cases, the adoption of open source software (OSS) components can be a way of overcoming problems related to COTS components, such as source code not being available, the impossibility of driving the evolution of the product, the obligation to upgrade the product and the need for the new version to conform to the wrap code already written (P182).

Real-time and embedded systems. CBSE also has prominence in embedded and real-time systems (P1136), addressed in 84 and 54 studies, respectively. In addition, distributed systems (independent, or as real-time and sometimes embedded systems) are the application domain in 46 studies. Such systems can be used in business domains such as: robotics, industrial control, automotive, automation systems, distributed wireless networks, air-traffic management, shipboard computing, etc.

The main concerns in these studies are non-functional properties – timing (response time, execution time, worst-case execution time), resources (memory and CPU consumption, energy savings), and questions relating to dependability (reliability, safety, availability, security). These systems have specific architecture and, often due to some specific solutions at run-time, they have different models during the design phase and at run-time (P67; P105; P115; P116; P171; P281; P387; P402; P417; P418; P419; P438; P441; P467; P494; P631; P812). At the same time CBSE imposes some disadvantages such as overhead in resource utilization, difficulties in achieving predictability and composability of important non-functional properties (P805).

AOSD. Aspect-Oriented Software Development is addressed by 23 studies. By employing CBSE, the implementation of certain concerns, such as logging, security and caching, cannot be confined into a single logical module. These concerns are called crosscutting as their implementation virtually crosscuts the traditional decomposition of a software application. AOSD provides a solution for modularizing these crosscutting concerns by introducing a new modularization entity, called an aspect (P1050).

SPL. This domain is addressed by 21 studies. Derivation of products in a product line is a transformation from common and varying requirements (or features) in the problem space to reusable components in the solution space of a product line (P694). Additionally, software product lines today can be built from components supplied by different vendors (P693).

Web-based systems. Today's enterprises increasingly rely on the Web to support their operations and integration of their business processes with those of their suppliers, partners, and customers (P473). These businesses must respond to changing business and competitive environments in near real-time. Web-based systems (addressed by 21 studies) implement software as components, and their functions appear as a set of services. In this context, change becomes a first-class design goal, requiring business and technology architecture whose software components can be added, modified, replaced, and reconfigured (P433).

SOA. Twenty primary studies propose the combination of software components and service-oriented architecture, since benefits from both areas are achieved in the same application. Services provide the advantages of abstraction as well as large-scale interoperability, and components are a robust approach based on the composition and reuse of clearly defined elements through their interfaces (P1183). In addition, a service has all the characteristics of a component and more: it can be developed using different technologies, and it can be executed independently in different run-time environments. In this paradigm shift, researchers and developers are moving from component-based to service-based development (P121). However, the component concept remains present since the core of service-oriented architectures (SOAs) is distributed software components provided or accessed by independent third parties (P18).

MDD. Model-driven development, addressed by 15 studies, can be employed in an efficient way in the design and implementation of component-based systems. According to (P121), MDD and CBSE can be considered as two orthogonal ways of reducing development complexity. The use of meta-models significantly reduces the time necessary to develop components and the supporting tools in both the design and run-time infrastructures (P655).

Operating systems. Eight studies applied CBSE in the operating systems domain. In particular, most studies address embedded operating systems. One of the issues in this field that is addressed by CBSE is dynamic reconfiguration, i.e. the ability to alter a system during its execution (P364; P654).

Summary - RQ4. In which domains has CBSE been applied?

The studies show that CBSE has been applied in many application and business domains. In many cases, the CBSE approach is used to address concerns specific to the domain and to enable a more efficient development process and better product quality. For example, in real-time embedded systems domains, functions implemented are quite simple, and thus the component interfaces are also simple. The emphasis is on non-functional properties (such as timing, resources) and the component models used in the domain are usually accompanied by time- and resource analysis. The main concerns in large distributed systems are complexity and evolution, which are supported by the dedicated component models for this domain. These component models include support for interface evolution, interaction protocols, and dynamic deployment. *Observation:* The domain support explains the large number of component models and technologies. While the basic principles such as interfaces, components, compositions, bindings, reusability, substitutability, etc. are the same, their realizations are quite different for different domains, which results from the main concerns of different domains.

Table 5
Research types (Wieringa et al., 2005).

Research Type	Description
Evaluation:	Techniques are implemented in practice and an evaluation of the technique is conducted. Implementation of the technique is shown in practice (solution implementation) and the consequences of the implementation in terms of benefits and drawbacks (implementation evaluation) are demonstrated.
Validation:	Techniques investigated are novel and have not yet been implemented in practice. The investigation uses a thorough, methodologically sound research setup, using for example experiments, prototyping, formal analysis, and similar.
Solution Proposal:	A solution for a problem is proposed, the solution can be either novel or a significant extension of an existing technique. The potential benefits and the applicability of the solution are shown by a small example, or demonstrator, or as a thorough line of argumentation.
Philosophical Argumentation:	These primary studies sketch a new way of looking at existing things by structuring the field in form of a taxonomy or conceptual framework.
Opinion:	These primary studies express opinions whether a certain technique is good or bad, or how things should been done, and similar. They do not rely on related work and research methodologies.
Experience Report:	Experience reports explain what and how something has been done in practice, for example case studies, surveys, and similar research from empirical software engineering.

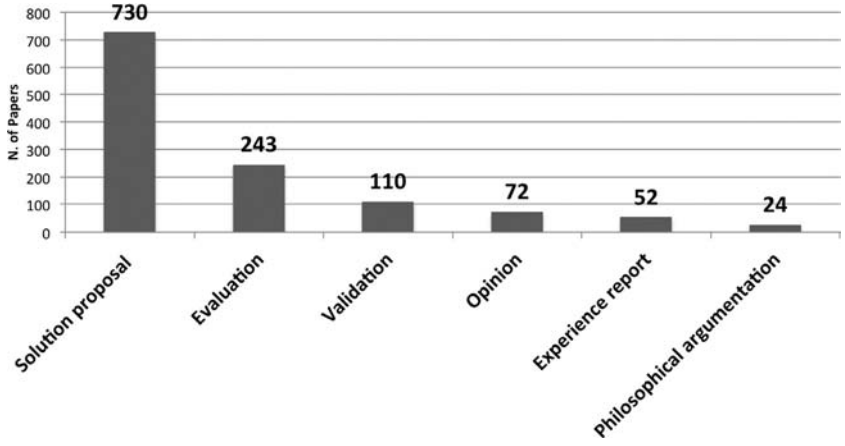

Fig. 12. Research type.

3.5. RQ5. What are the most frequently applied research types and methods? How have they changed over time?

This question investigates the research types addressed, empirical research methods and contribution types. According to our classification scheme presented in Table 5, the research types are: *validation, evaluation, solution proposal, philosophical argumentation, opinion*, and *experience report*.

According to the number of primary studies recorded (Fig. 12), these research types have not been addressed homogeneously. *Solution proposal* is the most frequently addressed type (59% of the total). However, the majority of *solution proposals* are evaluated or validated through small examples and in little detail, or through presenting arguments to show the feasibility and benefits of such a solution.

For *evaluation* research, addressed by 243 primary studies (20% of the total), the proposals are evaluated through case studies or surveys. *Validation* research is applied in 110 primary studies (8% of the total). These primary studies are validated through experiments or quasi-experiments. *Opinion* papers (72, 6% of the total) present discussions relating to the point-of-view of experts in the CBSE area. *Experience* papers (52, 4% of the total) describe the industrial experience in CBSE. The *philosophical* papers (24, 1% of the total) intend to structure concepts and definitions. Fig. 13 shows an evolution of research types over the years.

Empirical studies have become an important part of software engineering research and practice (Shull et al., 2007). Three empirical research methods applied in primary studies were identified: case study, survey and experiment. Case studies and surveys are applied in *evaluation* research primary studies; experiments (and quasi-experiments) are applied in *validation* research primary studies. Fig. 14 shows the applied empirical research methods.

The most frequently applied methods were case studies (228 primary studies, 18% of the total) and experiments (107 primary studies, 8% of the total). As a matter of fact, the case studies identified in the primary studies do not report proper details such as research questions, protocol, and data extraction techniques. This can make future replications difficult. This observation also holds for experiments, since primary studies do not present details such as subjects, hypotheses, or dependent and independent variables.

On the other hand, surveys (16 primary studies, 1% of the total) that are applied in *evaluation* research studies present an extensive analysis of the topic under study. In general, these studies follow a well-defined method with research questions or evaluation criteria, sample selection and questionnaire design. In contrast to the research methods mentioned above, some of the *solution proposals* present examples aimed at validating or evaluating their work. These examples differ in detail and structure across each primary study, however, all examples are used to present the strengths and drawbacks of the proposals.

Summary - RQ5. What are the most frequently applied research types and methods? How have they changed over time?

The studies presented show a clear tendency to provide "solutions" – i.e. new methods, processes, theories, formalisms, tools, etc. The number of case studies (18.5%) is not high, but it is significant. *Observation:* This shows that the CBSE area has reached a certain level of implementation and application in practice. However, further analysis of the case studies shows that many are demonstrators or studies of simple academic cases rather than industrial case studies.

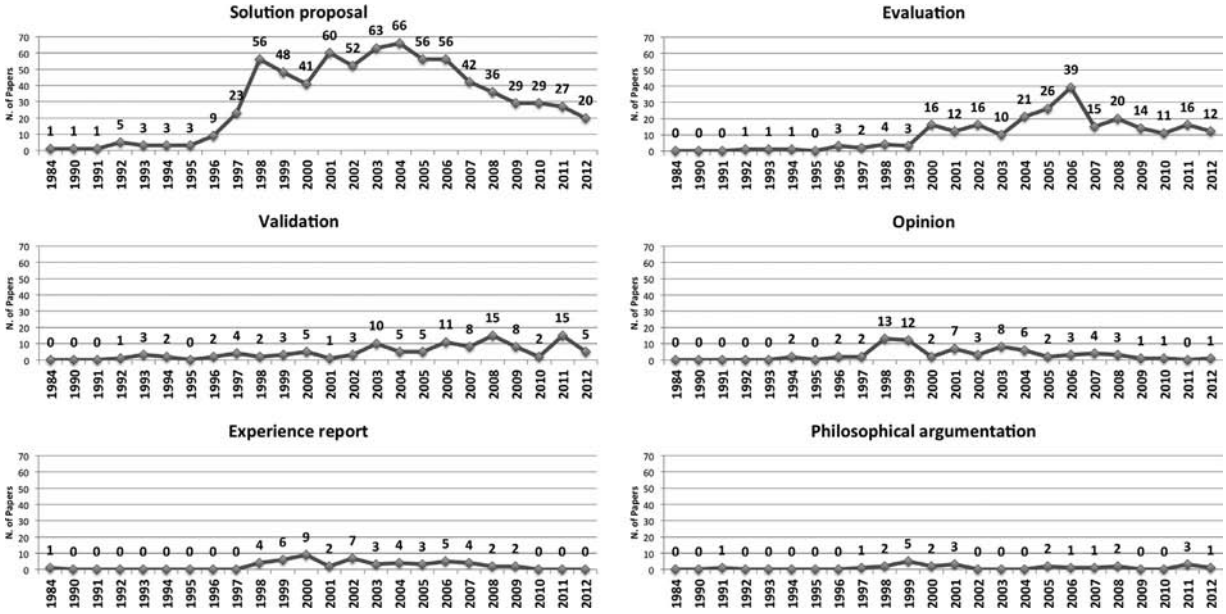

Fig. 13. Research type per year.

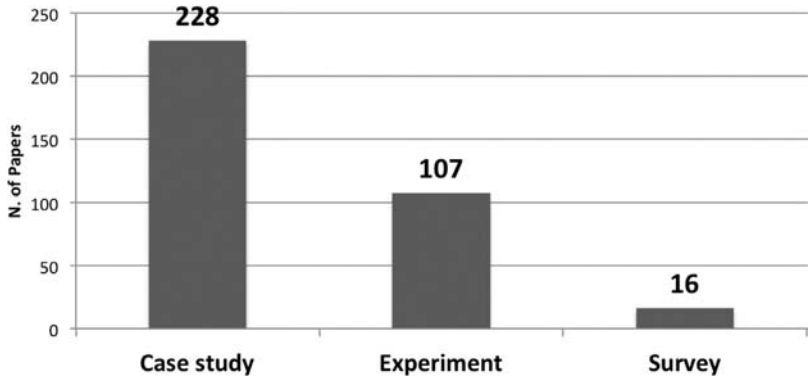

Fig. 14. Empirical research methods.

4. Gap analysis and topics for further research

The analysis of the primary studies showed the characteristics and their evolution through more than two decades. Here we analyze combinations of the results from the stated questions and investigate whether these combinations can indicate possible missing links in the research. Based on these results we identify possible future steps in CBSE research.

Fig. 15 summarizes the results from the three research questions posed in this mapping study: research topics (classified according to Table 4: *functionality, methodology, quality, management, interaction*, and *evolution and tools*), research types (according to Table 5: *solutions, evaluation, validation, philosophical argumentation, opinion, experience reports*) and contribution types (according to Table 6: *process, modeling, discussion, method, technology*, and *metrics*).

From Fig. 15 we can identify several gaps in the research performed – i.e. a difference between a given and new potential research. We refer to research topics in combination with (a) research types discussed in Section 4.1, (b) contribution types discussed in Section 4.2, and (c) the studies themselves classified in these categories, as specified in the repository[8]. Based on this analysis we propose topics for further research in Section 4.3.

[8] Available at http://tassiovale.com/JSS-ValeEtAl-DataExtraction.html (File size: 2.4MB)

4.1. Research topics vs. research types

We have already identified that the research type is dominated by new *solution proposals* (which is quite expected for research in a new area). Most of the new *solution proposals* address *functionality*; the second largest topic is *interaction* (i.e. interaction between components, and component compositions) which is at the heart of CBSE. Solutions for *quality* concerns are the third most common research type.

The research type *evaluation* (i.e. analysis of existing solutions) is ranked second, though considerably lower than *solution proposals*. *Evaluation* research is equally distributed among the research topics, approximately following the distribution for *solution proposals*. For both solution and evaluation we can see lower contribution in *management*-related research (gap *G1*).

A similar distribution exists for *validation* research, though it is less prevalent in the studies. However, a gap is visible from a small number of *validation* research in relation to *functionality*; While there are many proposed solutions, their validations are significantly less present. This indicates a gap between solutions and their validations in practice. (gap *G2*).

The *experience reports, philosophical argumentation* and *opinion* research types are considerably less frequently represented in comparison to *solution proposals, evaluation* and *validation*. While it is to be expected that *philosophical* and *opinion* research types are less prevalent, due to the nature of CBSE problems which are more related to

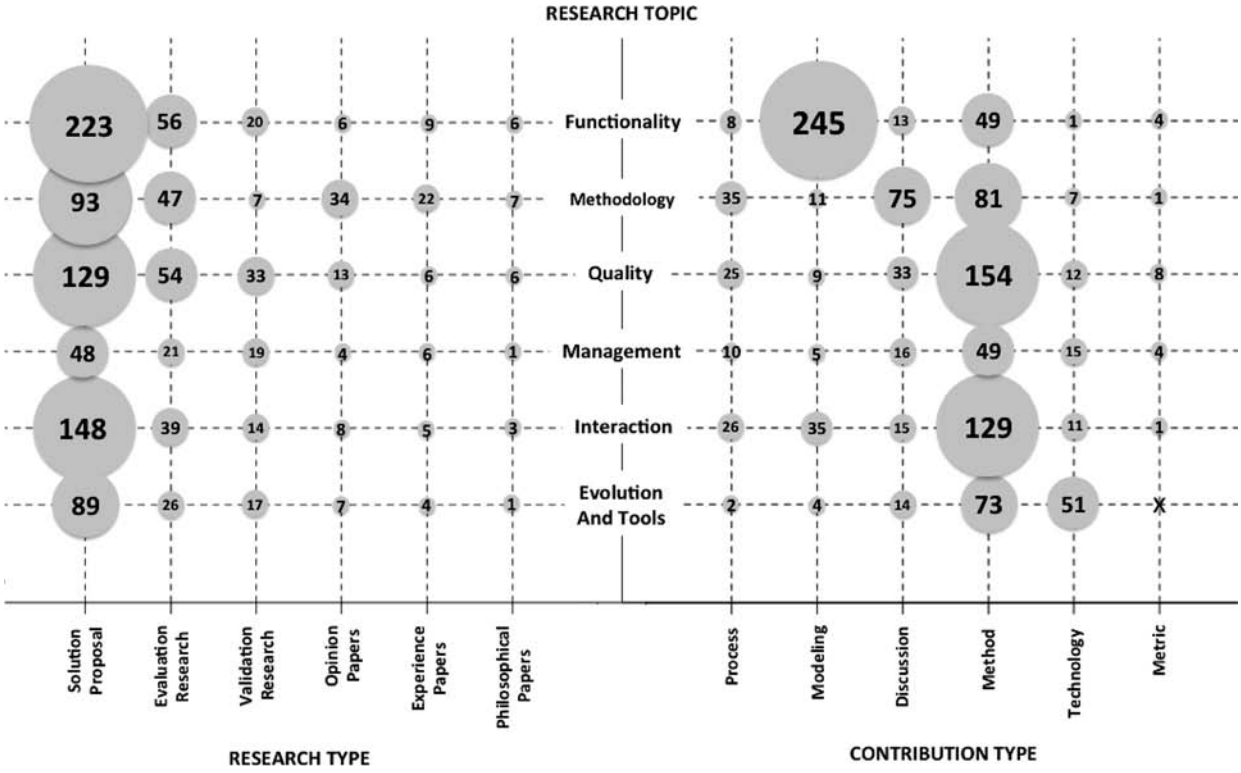

Fig. 15. CBSE evidence.

Table 6
Contribution types.

Contribution Type	Description
Method:	In this category, the contribution of the study can propose an algorithm, a mechanism, a theory, etc. to deal with different aspects of CBSE.
Process:	A methodology is proposed which describes rules, guidelines and a sequence of steps for how things should be performed, e.g., activity for creating and developing reusable assets, architecture, etc.
Modeling:	The contribution of the study can be either conceptual modeling/design for the problem under study or, novel or a significant extension of an existing modeling notation.
Technology:	A software tooling support or component platform is developed in order to support different aspects of the problem under study.
Discussion:	In this category, comparison of different characteristics of existing approaches or area overview are considered.
Metric:	This category proposes metrics to evaluate different CBSE aspects.

Table 7
List of authors with top 47 citations.

No	Author	Publ.	Citat.		Author	Publ.	Citat.
1	Basili, Victor	9	1331	25	Quema, Vivien	1	617
2	Coupaye, Thierry	6	1145	26	Lau, Kung-Kiu	20	612
3	Crnkovic, Ivica	23	1053	27	Visnovsky, Stanislav	2	539
4	van Ommering, Rob	4	1050	28	Brown, Alan W.	1	534
5	Bruneton, Eric	2	972	29	Neighbors, James	1	527
6	Leclercq, Matthieu	2	972	30	Chaki, Sagar	1	515
7	Batory, Don	5	855	31	Clarke, Edmund	1	515
8	Caldiera, Gianluigi	3	838	32	Groce, Alex	1	515
9	Kramer, Jeff	1	802	33	Jha, Somesh	1	515
10	Magee, Jeff	1	802	34	Veith, Helmut	1	515
11	van der Linden, Frank	2	802	35	Wang, Zheng	4	485
12	Jezequel, Jean-Marc	4	788	36	Notkin, David	2	460
13	Plouzeau, Noel	4	764	37	Ncube, Cornelius	4	449
14	O'Malley, Sean	1	730	38	Ball, Thomas	1	447
15	Johnson, Ralph	1	710	39	Nagappan, Nachiappan	1	447
16	Beugnard, Antoine	1	704	40	Zeller, Andreas	1	447
17	Watkins, Damien	1	704	41	Morisio, Maurizio	7	437
18	Stefani, Jean-Bernard	5	688	42	Gorton, Ian	10	417
19	Reussner, Ralf	9	681	43	Mezini, Mira	4	406
20	Voas, Jeffrey	6	679	44	Schmidt, Heinz	6	403
21	Zaremski, Moormann	1	676	45	Canal, Carlos	2	389
22	Wing, Jeannette	1	676	46	Becker, Steffen	5	384
23	Wallnau, Kurt	6	635	47	Cervantes, Humberto	5	382
24	Plasil, Frantisek	7	629				

Table 8
Research topics and subtopics.

Research topics	Subtopic
Functionality: (312 studies)	Component models (134 studies), software architecture (96 studies), component specifications (54 studies), languages (48 studies)
Interaction: (208 studies)	Component composition (150 studies), middleware (6 studies), contracts (17 studies), documentation (2 studies), algorithms (1 study)
Quality: (231 studies)	Quality in general (45 studies), testing (48 studies), verification (45 studies), performance (35 studies), reliability (17 studies), security (12 studies), fault management (9 studies), metrics (2 studies)
Management: (96 studies)	Certification (19 studies), configuration (22 studies), deployment (9 studies), repositories (8 studies), guidelines (1 study), component management in general
Evolution and Tools: (141 studies)	Development tools (60 studies), libraries (9 studies), adaptation (34 studies), updating (7 studies), evolution in general (11 studies), environments (6 studies), techniques (4 studies), technologies (3 studies), platforms (1 study), customization (1 study)
Methodology: (196 studies)	Search and retrieval (59 studies), design (29 studies), methodologies in general (4 studies), requirements elicitation (1 study)

practical solutions than principles questions of software engineering, it is indicative that the presence of *experience reports* is low. This characterizes a clear gap in CBSE research (gap *G3*). This also indicates that CBSE concepts are not fully established in industry, or that the CBSE concepts may be integrated into overall development and maintenance processes, but not identified as CBSE principles (gap *G4*). The lack of principles questions could also be the explanation for somewhat unclear concepts (e.g. what is a software component? What is a component model?), or for concepts that are difficult to integrate into the overall development process. Bosch (2002) claims that understanding components in industry is different from academia: while components in academia are understood as small, well-defined executable units (through their interface), in industry a component is understood as an executable unit, but more like a large package of software, most likely not formally defined (if defined at all). This may also be a reason for the small number of *experience reports* directly related to CBSE, calling for further research on how to deploy the CBSE approach in industry in an efficient way.

4.2. Research topics vs. contribution types

The *process* contribution type (see Fig. 15) is rather small (only 106 studies, or less than 10%) and this is a matter of tradition; CBSE has emerged from object-oriented language communities and software architecture communities in which modeling and methods are of greater interest than the processes. Consequently, relations between processes and different research topics (e.g. *functionality, methodology, management*) can be further developed – for example how does CBSE work in agile processes? How to increase efficiency in reuse processes? How to assess components?, etc.) (gap *G5*)

The *modeling* contribution type (with 309 studies, or 20%) is dominated by *functionality* (245 of 309 studies). The studies address component models, new or refined from existing component models, or component model specialization in particular domains. Other research topics are considerably less represented. While we can expect that *evolution and tools, management*, and *methodology* are less present because of weaker relations with modeling in general (e.g. modeling of tools, or management modeling), there is a surprisingly low number of studies in relation to *quality* modeling. This is in line with the findings from Crnkovic et al. (2011a) where the majority of component models, specially those used in industry, have poorer support for managing non-functional properties (gap *G6*).

Among all contribution types, the *method* type dominates (535 studies, or 43%) between the contribution types, and covers all kinds of research results, but mostly frequently *interactions* (together 217 studies, or 17%), and *quality*-related results (241 studies, or 19%). In these studies the research results are often built on existing research results from software engineering and applied to CBSE. For example, the known solutions for reliability in software engineering are applied specifically to CBSE by using rules defined by component models.

The other contribution types (*discussion, technology, metrics*) are considerably less represented (166 studies, or 14%, 97 or 8%, and 18 or less than 2%, respectively), which is in line with software engineering in general, as shown in Shaw (2003). *Metrics* are rarely used, which indicates that CBSE metrics are not yet fully developed (gap *G7*), which is related to a lack of formal definitions of components (gap *G8*). The *technology* contribution type is mostly underpinned by tools developed to support CBSE (gap *G9*). One of the interesting and yet unanswered questions is whether CBSE support can be more efficiently obtained by developing new, CBSE-specific tools, or by improving the existing tools being able to support CBSE.

4.3. Incentives for further research

Table 9 summarizes the identified gaps related to the research types.

From Table 9, Fig. 15, and the discussions in Sections 4.1 and 4.2 we have derived some directions in research that can advance CBSE state of the art and improve state of the practice. These are: *evidence and analysis of CBSE use in industry; new support for CBSE lifecycle activities.; improved formalisms.; development of CBSE-related tools.*; and *successful integration of CBSE in new areas*. A more detailed description of these directions is presented as follows.

- *Evidence and analysis of CBSE use in industry*.
  From state-of-practice (e.g use of component-based technologies in industry) it is observable that CBSE is widely used in industry. It is, however, not evident to what extent component-based principles are applied, and which parts of CBSE have been successfully applied and which not. In general there is a lack of evidence concerning industrial applications of CBSE, which is visible from the low number of *experience papers* as research types. This calls for different types of empirical studies of industrial CBSE applications:
  - *Empirical studies of use/benefits of CBSE in industry* (related to gaps *G1* and *G2*). There is a huge potential for CBSE application in different business areas, but each area has its own characteristic benefits and challenges. Empirical studies about CBSE use, benefits and associated challenges in different business domains can provide interesting answers for further improvements;
  - *CBSE deployment in industry* (related to gap *G3*). How can CBSE be successively introduced into a company's development process? To what extent should the development process be CBSE oriented? Which types of components (COTS, internally developed components or open-source components) should be

Table 9
Identified gaps.

Gap	Description	Gap type
G1	Lack of research of CBSE-related management.	Management vs. all research type
G2	Lack of validation in industrial context.	Solution vs. validation
G3	Lack of experience reports of CBSE use in practice.	Experience vs. research topics
G4	Lack of clear concepts of CBSE in practice.	Experience vs. research topics
G5	Lack of CBSE integration in lifecycle processes.	Process vs. research topics
G6	Lack of quality attributes modeling in CBSE.	Quality vs. modeling
G7	Lack of metrics related to CBSE.	Metrics vs. research topics
G8	Lack of formal definitions of components.	Metrics and modeling vs. quality
G9	Lack of CBSE tool support.	Tools vs. process & modeling.

reused? Does the CBSE deployment require organizational and managerial changes? Answers to these and similar questions are still not widely known and need further investigation.

  - *Empirical studies about tool implementation and tool usage* (related to gap *G9*). To make CBSE approach efficient, significant automation of the development process must be introduced (Szyperski, 2002; Schneider and Han, 2004; Crnkovic, 2001). That requires substantial investment in tools - both in the existing tools that implement particular technologies, and in development of new tools for automation of different activities in the development process, and in particular automation of information transfer and transformation between different lifecycle phases (for example generating search for components from requirement specifications, or test generation from component specifications). Which tools are the most beneficial, which are needed, and which should be improved? Those are some of the questions that should be addressed by empirical studies about tool usage in industrial contexts.

- *New support for CBSE lifecycle activities* (related to gaps *G3* and *G5*). Compared to other research topics (see Fig. 15, *Process* contribution type), there are considerably fewer results relating to the CBSE development process and lifecycle models related to CBSE. In particular there are not many studies about how the new agile methods should be applied to component-based development. Does CBSE require some specific activities in agile development, for example component testing or refactoring? How can CBSE improve software evolution and continuous software deployment? What is the relationship between software component evolution and software system evolution? Such questions require further research in order to take advantage of CBSE in modern development processes.
- *Improved formalisms.*
  CBSE includes a number of well-defined formalisms (Fig. 15, *Modeling* contribution type), which enables automation and thereby more efficient development and high-quality software. These formalisms are mainly related to component specification (interface) and their functional compositions. There are however areas in which formalisms have not been fully developed and implemented. There is a lack of:
  - *Improved formalisms for non-functional properties in components* (related to gaps *G6* and *G8*). Non-functional properties (Fig. 15, *Modeling, Quality* research topic) address a very large number of components, system requirements and characteristics, and their importance depends on the business and engineering domains. While we can find developed theories and formalisms in analysis of certain non-functional properties (e.g. resource-usage in resource-constraint systems, timing properties in real-time systems, or variability in product lines), applications of these theories on components and component-based systems still need further development.
  - *Improved formalisms for component behavior – i.e. dynamic aspects of interactions* (related to gaps *G7* and *G8*). Component specifications in the form of interfaces defining signatures and possibly their semantics are more developed than dynamic aspects of component interactions, or the dynamic behavior of a component. Further development of these topics will lead to improved analysis and increased automation of component compositions.
  - *CBSE metrics – formalization and empirical studies* (related to gaps *G4* and *G7*). Specific metrics (Fig. 15, *Metrics* contribution type) for CBSE are not identified. What is a good use of CBSE? How are CBSE principles well utilized in specific domains? These are the questions that need further research. These questions are related to the number of metrics for CBSE that need to be developed.
- *Development of CBSE-related tools* (related to gap *G9*). There exists many tools that include a term “component” and some operations related to “components” (for example UML tools, different ADL tools, Simulink, etc.). In these tools components are architectural units. Additional features, such as component attributes compositions, reusability, component adaptions, etc. are missing. This gives opportunities to enrich different tools with the basic features of CBSE.
- *Successful integration of CBSE in new areas.*
  Fig. 2 shows the trend of a decreasing number of CBSE-related publications in the last five years (2007–2012). The use of components is, however, increasing (in the form of the reuse of standard components, services, infrastructural components, open source, etc.). In recent years new trends in software development have started to receive attention: cloud computing, real-time big data processing, machine learning, use of heterogeneous computing units, ubiquitous computing – these are some of the new areas that require new architectural solutions, new communication patterns, etc. The demand for efficiency leads to the usage of components in such areas, which calls for research into how to optimally apply the existing component models, or which type of adaptations might be useful, and finally whether some new component models are desirable. Requirements for management of particular non-functional properties (e.g. finding components for specific services, ensuring quality of services like response time or scalability, etc.) may appear. When developing systems running on heterogeneous platforms (i.e. different executable units) a need for different component implementations arises. This requires efficient development of components with different implementations. Adaptation requirements put new demands on component adaptations that go beyond the use of adapters and/or wrappers.

5. Threats to validity

In empirical software engineering, validity threats are categorized under four distinct concerns Petersen et al. (2008); Perry et al. (2000):

- *Construct validity* relates to the collected data and how well the data represent the investigated phenomenon.

- *Internal validity* concerns the connection between the observed behavior and the proposed explanation for the behavior, i.e., it is about ensuring that the actual conclusions are true.
- *External validity* concerns the possibility of generalizing the results of a study.
- *Reliability* concerns the possibility of reaching the same conclusions if the study was repeated by another researcher.

We discussed these concerns below.

5.1. Construct validity

In this study, *construct validity* relates to the questions of whether all important sources have been investigated, and whether non-relevant sources have been taken into consideration. This includes:

- **Selected conferences and journals:** Not all potential journals and conferences were included in the manual search. Our selection criterion was to consider only conferences and journals presenting a high percentage of CBSE primary studies, analyzing an important set of representative studies. In order to determine a reliable list, we had several discussion meetings with the project members. We also analyzed backward references, i.e., look at the references of the primary studies. A few new primary studies were identified in this way.
- **Inclusion criteria:** The decision of whether or not to include a candidate study was based primarily on reading the title and abstract of each study. Thus, some studies may have been erroneously included or excluded. The mitigation strategy was to perform it using peer review and applying the exclusion criteria on introduction and conclusion of the resulting candidate studies from the first iteration.
- **Publication bias:** Regarding the publication bias, we cannot guarantee that all relevant primary studies were selected, since only a manual search was performed. Only primary studies that answered our research questions were included. By selecting the most relevant journals and conferences, this risk was minimized. Checking backward references minimized the risk of omitting some important sources.

5.2. Internal validity

Internal validity in our research concerns the risk of whether our methods could lead to wrong or irrelevant conclusions. The following activities could threaten internal validity.

- **Research questions:** The set of questions we defined might not have covered the whole CBSE area, which implies that one may not find answers to the questions that concern them. As we considered this a feasible threat, we had several discussion meetings with project members and experts in the area aiming to calibrate the questions. Furthermore, we analyzed whether a particular study addressed concerns that were outside of our questioning (i.e. whether a study could not be categorized in respect to a research question), which was not the case.
- **Studies classification:** During the data extraction process, the studies were classified based on our judgment. In order to mitigate this threat, the classification process was performed by peer review in several iterations.
- **Studies validation/evaluation:** We did not analyze the research rigor applied in the studies validation/evaluation. However, we performed quality assessment of our selection: Only the studies from peer-reviewed events and journals were selected. Furthermore, we identified the studies with higher impact (i.e. a higher citation number and published in journals and proceedings of higher impact).

5.3. External validity

This concern is covered in our observations and conclusions. However, a mapping study does not aim to generalize the findings outside the subject of the research (in this case CBSE), and for this reason external validity is not relevant as a threat. Some of the questions, and the results, could certainly be applied to other approaches or technologies in software engineering, but this concern is out of the scope of this research.

5.4. Reliability

The research protocol specification and its strict realization make it possible to repeat the first part of the selection process (from the selected sources). This part is fully repeatable. Further selection, based on assessments by reading the primary studies is part of a subjective process and may lead to different selection and classification of the studies, but this part includes research creativity, and belongs to the research contribution.

6. Related work

The literature on CBSE provides a large number of studies, regarding both general and specific issues. Amongst them, we have identified some works developed to gather and evaluate available evidence in the area. They are thus considered to have similar ideas to our mapping study and are described below.

Brown and Wallnau (1998) present a report about some discussion points from a workshop on Component-Based Software Engineering (CBSE), providing a useful synthesis of participants' diverse perspectives and experiences. The results suggested that CBSE is a coherent engineering practice, and there has been good progress in identifying its core concepts, as well as different perspectives on them. The discussion involved conceptual (what is the core part of CBSE?), skeptical (why will it work now if it did not before?), and practical (what will it mean to organisations if CBSE does work?) aspects. This early work was a very important analysis in the starting period of extensive CBSE research, but comparing to our mapping study does not include evidence and state-of-the-art based on such evidence.

Crnkovic (2001) provided an overview of current CBSE challenges at that time, and some principle disadvantages that make CBSE deployment more difficult. Examples of these difficulties are time and effort required for developing components, unclear and ambiguous requirements, conflict between usability and reusability, and component maintenance costs. Most of these disadvantages have been addressed and have decreased in the years since. Similarly to Brown and Wallnau (1998) this work is rather a list of challenges and a source of inspiration for further research than evidence-based analysis.

Schneider and Han (2004) reviewed some of the component-based development goals, and proposed thoughts and discussions about the next decade of component technology. The work investigated whether the issues stated previously had been addressed. Furthermore, the authors investigated what remains to be done and where the efforts should be focused to improve component technologies, propose topics for further investigation such as functionality, interaction, quality, management, evolution/tools and methodology. The work is a result of an analysis of CBSE in its exploration and start of implementation phase, but it does not provide a complete analysis of the state-of-the-art and state-of-the-practice.

Mahmood et al. (2007) presented a literature survey about proposed techniques for the CBSE lifecycle phases. Basically, this work considers the following activities for CBSE: component identification and selection; component integration; deployment; and component evolution. This study aimed to provide a better understanding of the

existing different techniques for each area. Although the aforementioned studies present a literature review, none of them were performed in a systematic way. Further the emphasis is on software lifecycle, while other aspects are omitted.

Maras et al. (2012) analyze 15 years of activities of the ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE). They apply systematic review guidelines specifically for papers published in the CBSE Symposium to investigate four stated research questions concerning the impact of publications, research topics, reported questions, results and research methods. The research topics identified there are component models, component technologies, extra-functional properties, composition & predictability, software architecture, quality, lifecycle, domains and methodology. The topics identified in these events are somewhat different than those defined in Schneider and Han (2004) and used in this paper, but there is a clear correspondence between them.

An overview of the CBSE objectives and their feasibilities is discussed in Almeida et al. (2007), in which it is claimed that software components may be the most appropriate solution for real-world systems, considering that, instead of replacing the whole system, "pieces" of software or components may be added, removed or replaced, attending to new possible requirements, which can reduce time-to-market and improve product quality.

Crnkovic et al. (2011b) provide a short retrospective of CBSE, the accumulated experience in using CBSE, the remaining challenges, and the opportunities to increase use of CBSE in different software engineering areas and domains (e.g. model-driven engineering and software product lines).

Despite most studies (Brown and Wallnau, 1998; Crnkovic, 2001; Schneider and Han, 2004; Mahmood et al., 2007; Almeida et al., 2007; Crnkovic et al., 2011b) propose overviews or discussion points on the CBSE field, they do not apply systematic techniques to analyze the papers. In addition, they do not select a representative sample of CBSE primary studies if compared with this study. Maras et al. (2012) apply systematic review techniques to synthesize evidence on CBSE studies, however, they focus on a specific venue, the CBSE Symposium. The contributions from this related work are partially and indirectly included in our work, when their findings correspond to the evidences we provided for the literature review.

7. Concluding remarks

In this paper, we performed a literature review to investigate the state- of-the-art in the CBSE area. As the result of a manual search, 1396 primary studies were identified, of which 1231 were considered relevant as primary studies. Five aspects of these studies were analyzed: main objectives, research topics, application domains, research intensity and research types and methods.

Increasing productivity, cost savings and increasing quality are the most frequently mentioned CBSE objectives. In order to achieve these objectives, six research topics were homogeneously addressed, with greater attention devoted to *functionality* issues, such as component models, specification techniques, properties and component-based architectures.

In addition, software components are used in several application domains. CBSE has become part of some of these domains. The first one is COTS, the domain with the highest number of studies, due to the synergy between these two areas where most COTS systems are supported by off-the-shelf components. COTS is followed by embedded, real-time and distributed systems as the most frequently addressed domains.

When considering research types, the *solution proposal* is prevalent. *Solution proposals* are followed by *evaluation research* and *validation research*. Case studies and experiments are the most frequently applied research methods, however, the rigor with which research is conducted should be investigated.

This work forms the basis for identifying gaps in research which should be addressed, and for pinpointing areas where further investigation of the existing studies should be performed. As future work, we intend to analyze the fifty most-cited publications (including books and technical reports) in the CBSE field. The goal is to extract information about our research questions as well as to perform a detailed analysis of specific topics, and probably to make a comparison with the available evidence gathered for this work (Table A.1).

Acknowledgment

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES[9]), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08, by the project RALF3, funded by the Swedish Foundation for Strategic Research, and project ORION funded by The Knowledge Foundation, Sweden. We would also like to thank Josip Maras who automated collection of some data.

Appendix A. The most cited primary studies

[9] http://www.ines.org.br

Table A.1
Most cited primary studies.

Index	ID	Primary study title	Citations
1	P16	R. van Ommering, F. van der Linden, J. Kramer, J. Magee, The koala component model for consumer electronics software, IEEE Computer 33 (2000) 78–85.	802
2	P415	D. Batory, S. O'Malley, The design and implementation of hierarchical software systems with reusable components, ACM Transactions on Software Engineering and Methodology 1 (1992) 355–398.	730
3	P423	R.E. Johnson, Frameworks = (components + patterns), Communications of the ACM 40 (1997) 39–42.	710
4	P10	A. Beugnard, J.M. Jézéquel, N. Plouzeau, D. Watkins, Making components contract aware, IEEE Computer 32 (1999) 38–45.	704
5	P418	A.M. Zaremski, J.M. Wing, Specification matching of software components, ACM Transactions on Software Engineering and Methodology 6 (1997) 333–369.	676
6	P35	E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, J.B. Stefani, The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems, Software Practice & Experience 36 (2006) 1257–1284.	617
7	P1216	A.W. Brown, K.C. Wallnau, The current state of cbse, IEEE Software 15 (1998) 37–46.	534
8	P623	J.M. Neighbors, The draco approach to constructing software from reusable components, IEEE Transactions on Software Engineering 10 (1984) 564–574.	527
9	P235	S. Chaki, E.M. Clarke, A. Groce, S. Jha, H. Veith, Modular verification of software components in c, IEEE Transactions on Software Engineering 30 (2004) 388–402.	515

(*continued on next page*)

Table A.1 (*continued*)

Index	ID	Primary study title	Citations
10	P632	F. Plasil, S. Visnovsky, Behavior protocols for software components, IEEE Transactions on Software Engineering 28 (2002) 1056–1076.	465
11	P241	N. Nagappan, T. Ball, A. Zeller, Mining metrics to predict component failures, in: Proceedings of the 28th International Conference on Software Engineering, ICSE '06, ACM, New York, NY, USA, 2006, pp. 452–461.	447
12	P979	E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, J.B. Stefani, An open component model and its support in java, in: I. Crnkovic, J. Stafford, H. Schmidt, K. Wallnau (Eds.), Component-Based Software Engineering, Vol. 3054 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 7–22.	355
13	P1	G. Caldiera, V.R. Basili, Identifying and qualifying reusable software components, IEEE Computer 24 (1991) 61–70.	340
14	P535	S. Becker, H. Koziolek, R. Reussner, The palladio component model for model-driven performance prediction, Journal of Systems and Software 82 (2009) 3–22.	334
15	P548	E.J. Weyuker, Testing component-based software: A cautionary tale, IEEE Software 15 (1998) 54–59.	330
16	P767	E. Bruneton, T. Coupaye, J.B. Stefani, Recursive and dynamic software composition with sharing, in: Seventh International Workshop on Component-Oriented Programming (WCOP 2002), 2002.	315
17	P628	W.B. Frakes, T.P. Pole, An empirical study of representation methods for reusable software components, IEEE Transactions on Software Engineering 20 (1994) 617–630.	
18	P638	K.K. Lau, Z. Wang, Software component models, IEEE Transactions on Software Engineering 33 (2007) 709–724.	299
19	P553	N.A. Maiden, C. Ncube, Acquiring cots software selection requirements, IEEE Software 15 (1998) 46–56.	285
20	P206	J. Kontio, A case study in applying a systematic method for cots selection, in: Proceedings of the 18th International Conference on Software Engineering, ICSE '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 201–209.	272
21	P402	M. VanHilst, D. Notkin, Using role components in implement collaboration-based designs, in: Proceedings of the 11th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '96, ACM, New York, NY, USA, 1996, pp. 359–369.	270
22	P510	A. Bracciali, A. Brogi, C. Canal, A formal approach to component adaptation, Journal of Systems and Software 74 (2005) 45–54.	269
23	P500	R.H. Reussner, H.W. Schmidt, I.H. Poernomo, Reliability prediction for component-based software architectures, Journal of Systems and Software 66 (2003) 241–252.	262
24	P403	M. Mezini, K. Lieberherr, Adaptive plug-and-play components for evolutionary software development, in: Proceedings of the 13th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '98, ACM, New York, NY, USA, 1998, pp. 97–116.	243
25	P413	E. Ostertag, J. Hendler, R.P. Díaz, C. Braun, Computing similarity in a reuse library system: An ai-based approach, ACM Transactions on Software Engineering and Methodology 1 (1992) 205–228.	234
26	P625	R.S. Freedman, Testability of software components, IEEE Transactions on Software Engineering 17 (1991) 553–564.	229
27	P53	C. Atkinson, J. Bayer, D. Muthig, Component-based product line development: The kobra approach, in: Proceedings of the First Conference on Software Product Lines : Experience and Research Directions: Experience and Research Directions, Kluwer Academic Publishers, Norwell, MA, USA, 2000, pp. 289–309.	228
28	P1232	I. Crnkovic, Component-based software engineering - new challenges in software development, in: Proceedings of the 25th International Conference on Information Technology Interfaces, 2003, pp. 9–18.	224
29	P428	D. Sprott, Enterprise resource planning: Componentizing the enterprise application packages, Communications of the ACM 43 (2000) 63–69.	220
30	P430	J. Hopkins, Component primer, Communications of the ACM 43 (2000) 27–30.	216
31	P407	M. Odersky, M. Zenger, Scalable component abstractions, in: Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '05, ACM, New York, NY, USA, 2005, pp. 41–57.	215
32	P231	J. Hatcliff, X. Deng, M.B. Dwyer, G. Jung, V.P. Ranganath, Cadena: An integrated development, analysis, and verification environment for component-based systems, in: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 160–173.	213
33	P486	J.M. Voas, Certifying off-the-shelf software components, IEEE Computer 31 (1998) 53–59.	213
34	P226	R. van Ommering, Building product populations with software components, in: Proceedings of the 24th International Conference on Software Engineering, ICSE '02, ACM, New York, NY, USA, 2002, pp. 255–265.	210
35	P239	H. Cervantes, R.S. Hall, Autonomous adaptation to dynamic availability using a service-oriented component model, in: Proceedings of the 26th International Conference on Software Engineering, ICSE '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 614–623.	208
36	P405	S. McDirmid, M. Flatt, W.C. Hsieh, Jiazzi: New-age components for old-fasioned java, in: Proceedings of the 16th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '01, ACM, New York, NY, USA, 2001, pp. 211–222.	208
37	P9	B. Boehm, C. Abts, Cots integration: Plug and pray?, IEEE Computer 32 (1999) 135–138.	204
38	P238	C. Szyperski, Component technology: What, where, and how?, in: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 684–693.	201
39	P456	J. Bosch, Superimposition: a component adaptation technique, Information and Software Technology 41 (1999) 257 – 273.	198
40	P414	K.J. Sullivan, D. Notkin, Reconciling environment integration and software evolution, ACM Transactions on Software Engineering and Methodology 1 (1992) 229–268.	190
41	P222	D. Hamlet, D. Mason, D. Woit, Theory of software reliability based on components, in: Proceedings of the 23rd International Conference on Software Engineering, ICSE '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 361–370.	182
42	P237	B. Meyer, The grand challenge of trusted components, in: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, IEEE Computer Society, Washington, DC, USA, 2003, pp. 660–667.	180
43	P1193	D. Giannakopoulou, C.S. P"s"reanu, H. Barringer, Assumption generation for software component verification. In: Proceedings of the 17th IEEE International Conference on Automated Software Engineering, ASE '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 3–.	176
44	P2	R. Adler, Emerging standards for component software, IEEE Comput. 28 (1995) 68–77.	175
45	P626	L. Briand, V. Brasili, C. Hetmanski, Developing interpretable models with optimized set reduction for identifying high-risk software components, IEEE Trans. Softw. Eng. 19 (1993) 1028–1044.	174
46	P209	R.B. Kieburtz, L. McKinney, J.M. Bell, J. Hook, A. Kotov, J. Lewis, D.P. Oliva, T. Sheard, I. Smith, L. Walton, A software engineering experiment in software component generation. In: Proceedings of the 18th International Conference on Software Engineering, ICSE '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 542–552.	167
47	P546	W. Kozaczynski, G. Booch, Guest editors' introduction: Component-based software engineering, IEEE Softw. 15 (1998) 34–36.	163
48	P412	V.R. Basili, G. Caldiera, G. Cantone, A reference architecture for the component factory, ACM Trans. Softw. Eng. Methodol. 1 (1992) 53–80.	160

(*continued on next page*)

Table A.1 (*continued*)

Index	ID	Primary study title	Citations
49	P624	D.E. Harms, B.W. Weide, Copying and swapping: Influences on the design of reusable software components, IEEE Trans. Softw. Eng. 17 (1991) 424–435.	158
50	P15	J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, J. Phillips, N. Jackiw, D. Suthers, Developing educational software components, IEEE Comput. 32 (1999) 50–58.	154
51	P417	S. Henninger, An evolutionary approach to constructing effective software reuse repositories, ACM Trans. Softw. Eng. Methodol. 6 (1997) 111–140.	154
52	P439	K. Levi, A. Arsanjani, A goal-driven approach to enterprise component identification and specification, Commun. ACM 45 (2002) 45–52.	152
53	P217	M. Morisio, C.B. Seaman, A.T. Parra, V.R. Basili, S.E. Kraft, S.E. Condon, Investigating and improving a cots-based software development. In: Proceedings of the 22nd International Conference on Software Engineering, ICSE '00, ACM, New York, NY, USA, 2000, pp. 32–41.	150
54	P445	T. Ravichandran, M.A. Rothenberger, Software reuse strategies and component markets, Commun. ACM 46 (2003) 109–114.	147
55	P629	A. Arora, S. Kulkarni, Component based design of multitolerant systems, IEEE Trans. Softw. Eng. 24 (1998) 63–78.	146
56	P13	P. Narasimban, L. Moser, P. Melliar-Smith, Using interceptors to enhance corba, IEEE Comput. 32 (1999) 62–68.	144
57	P543	L. Hatton, Reexamining the fault density-component size connection, IEEE Softw. 14 (1997) 89–97.	142
58	P73	T. Biggerstaff, The library scaling problem and the limits of concrete component reuse. In: Proceedings of the Third International Conference on Software Reuse: Advances in Software Reusability, 1994, pp. 102–109.	139
59	P433	P. Fingar, Component-based frameworks for e-commerce, Commun. ACM 43 (2000) 61–67.	138
60	P5	D. Krieger, R. Adler, The emergence of distributed component platforms, IEEE Comput. 31 (1998) 43–53.	134
61	P220	E. Truyen, B. Vanhaute, B.N. Jørgensen, W. Joosen, P. Verbaeton, Dynamic and selective combination of extensions in component-based applications. In: Proceedings of the 23rd International Conference on Software Engineering, ICSE '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 233–242.	130
62	P416	A. Podgurski, L. Pierce, Retrieving reusable software by sampling behavior, ACM Transactions on Software Engineering and Methodology 2 (1993) 286–303.	130
63	P493	K.E. Emam, S. Benlarbi, N. Goel, S.N. Rai, Comparing case-based reasoning classifiers for predicting high risk software components, Journal of Systems and Software 55 (2001) 301–320.	130
64	P333	H. Hansson, M. Akerholm, I. Crnkovic, M. Torngren, Saveccm - a component model for safety-critical real-time systems, in: Proceedings of the 30th EUROMICRO Conference, EUROMICRO '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 627–635.	128
65	P635	K. Inoue, R. Yokomori, T. Yamamoto, M. Matsushita, S. Kusumoto, Ranking significance of software components based on use relations, IEEE Transactions on Software Engineering 31 (2005) 213–225.	128
66	P965	H. Cervantes, R.S. Hall, Automating service dependency management in a service-oriented component model, in: Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering, 2003.	126
67	P60	K. Czarnecki, U.W. Eisenecker, Components and generative programming (invited paper), in: Proceedings of the 7th European Software Engineering Conference Held Jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ESEC/FSE-7, Springer-Verlag, London, UK, UK, 1999, pp. 2–19.	125
68	P444	P. Vitharana, Risks and challenges of component-based software development, Communications of the ACM 46 (2003) 67–72.	125
69	P637	S. Kounev, Performance modeling and evaluation of distributed component-based systems using queueing petri nets, IEEE Transactions on Software Engineering 32 (2006) 486–502.	125
70	P1141	C. Ncube, N.A. Maiden, Pore: Procurement-oriented requirements engineering method for the component-based systems engineering development paradigm, in: International Workshop on Component-Based Software Engineering, 1999, pp. 1–12.	125
71	P211	R.K. Keller, R. Schauer, Design components: Toward software composition at the design level, in: Proceedings of the 20th International Conference on Software Engineering, ICSE '98, IEEE Computer Society, Washington, DC, USA, 1998, pp. 302–311.	124
72	P437	I. Crnkovic, B. Hnich, T. Jonsson, Z. Kiziltan, Specification, implementation, and deployment of components, Communications of the ACM 45 (2002) 35–40.	124
73	P87	M.L. Griss, Implementing product-line features with component reuse. In: Proceedings of the 6th International Conference on Software Reuse: Advances in Software Reusability, ICSR-6, Springer-Verlag, London, UK, UK, 2000, pp. 137–152.	122
74	254	M. Pinto, L. Fuentes, J.M. Troya, Daop-adl: An architecture description language for dynamic component and aspect-based development. In: Proceedings of the 2nd International Conference on Generative Programming and Component Engineering, GPCE '03, Springer-Verlag New York, Inc., New York, NY, USA, 2003, pp. 118–137.	121
75	P640	C. Canal, P. Poizat, G. Salaün, Model-based adaptation of behavioral mismatching components, IEEE Trans. Softw. Eng. 34 (2008) 546–563.	120
76	P248	L. Cheung, R. Roshandel, N. Medvidovic, L. Golubchik, Early prediction of software component reliability. In: Proceedings of the 30th International Conference on Software Engineering, ICSE '08, ACM, New York, NY, USA, 2008, pp. 111–120.	119
77	P406	M. Mezini, K. Ostermann, Integrating independent components with on-demand remodularization. In: Proceedings of the 17th ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, OOPSLA '02, ACM, New York, NY, USA, 2002, pp. 52–67.	118
78	P441	J. Sutherland, W.J. van den Heuvel, Enterprise application integration and complex adaptive systems, Commun. ACM 45 (2002) 59–64.	116
79	P1087	S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, I. Crnkovic, A component model for control-intensive distributed embedded systems., In: M. Chaudron, C. Szyperski, R. Reussner (Eds.), Component-Based Software Engineering, Vol. 5282 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 310–317.	116
80	P432	M. Sparling, Lessons learned through six years of component-based development, Commun. ACM 43 (2000) 47–53.	112
81	P442	A. Gokhale, D.C. Schmidt, B. Natarajan, N. Wang, Applying model-integrated computing to component middleware and enterprise applications, Commun. ACM 45 (2002) 65–70.	112
82	P522	M. kerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Håkansson, A. Möller, P. Pettersson, M. Tivoli, The save approach to component-based development of vehicular systems, J. Syst. Softw. 80 (2007) 655–667.	111
83	P431	C. Kobryn, Modeling components and frameworks with uml, Commun. ACM 43 (2000) 31–38.	108
84	P447	A.I. Mørch, G. Stevens, M. Won, M. Klann, Y. Dittrich, V. Wulf, Component-based technologies for end-user development, Commun. ACM 47 (2004) 59–62.	106
85	P552	J. Voas, Cots software: The economical choice?, IEEE Softw. 15 (1998) 16–19.	106
86	P61	M. Hauswirth, M. Jazayeri, A component and communication model for push systems., In: Proceedings of the 7th European Software Engineering Conference Held Jointly with the 7th ACM SIGSOFT International Symposium on Foundations of Software Engineering, ESEC/FSE-7, Springer-Verlag, London, UK, UK, 1999, pp. 20–38.	103
87	P156	Y. Wu, M.H. Chen, J. Offutt, Uml-based integration testing for component-based software. In: Proceedings of the Second International Conference on COTS-Based Software Systems, ICCBSS '03, Springer-Verlag, London, UK, UK, 2003, pp. 251–260.	103
88	P653	I. Crnkovic, S. Sentilles, A. Vulgarakis, M.R.V. Chaudron, A classification framework for software component models, Softw. Eng., IEEE Trans. 37 (2011) 593–615.	102

References

Almeida, E.S., Alvaro, A., Garcia, V.C., Mascena, J.C.C.P., Burgio, V.A.A., Nascimento, L.M., Lucrdio, D., Meira, S.R.L., 2007. C.R.U.I.S.E: Component Reuse in Software Engineering. C.E.S.A.R e-book.

Bosch, J., 2002. Architecture-centric software engineering. Proceedings of the 24th International Conference on Software Engineering, ICSE '02. ACM, New York, NY, USA, pp. 681–682. http://doi.acm.org/10.1145/581339.581443

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M., 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. 80, 571–583.

Brown, A.W., Wallnau, K.C., 1998. The current state of cbse. IEEE Softw. 15, 37–46.

Crnkovic, I., 2001. Component-based software engineering - new challenges in software development. Softw. Focus 2, 127–133. http://dx.doi.org/10.1002/swf.45

Crnkovic, I., Larsson, M., 2002. Building Reliable Component-Based Software Systems. Artech House publisher.

Crnkovic, I., Sentilles, S., Aneta, V., Chaudron, M.R.V., 2011a. A classification framework for software component models. IEEE Trans. Softw. Eng. 37, 593–615.

Crnkovic, I., Stafford, J., Szyperski, C., 2011b. Software components beyond programming: from routines to services, Software. IEEE 28, 22–26.

Heineman, G.T., Councill, W.T., 2001. Component-based software engineering: putting the pieces together. Addison-Wesley Professional.

Jorgensen, M., Shepperd, M., 2007. A systematic review of software development cost estimation studies. IEEE Trans. Softw. Eng. 33, 33–53.

Kitchenham, B., Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report Tech. rep. ebse 2007-001.

Lau, K.K., Ornaghi, M., Wang, Z., 2006. A software component model and its preliminary formalisation. Proceedings of the 4th International Conference on Formal Methods for Components and Objects, FMCO'05. Springer-Verlag, Berlin, Heidelberg, pp. 1–21.

Mahmood, S., Lai, R., Kim, Y., 2007. Survey of component-based software development. IET Softw. 1, 57–66.

Maras, J., Lednicki, L., Crnkovic, I., 2012. 15 years of cbse symposium: impact on the research community. In: International Symposium on Component-Based Software Engineering (CBSE), CBSE '12, pp. 61–70.

McIlroy, D., 1968. Mass-produced software components. In: NATO Conference, pp. 88–98.

Morisio, M., Torchiano, M., 2002. Definition and classification of cots: a proposal. International Conference on COTS-Based Software Systems (ICCBSS), ICCBSS '02. Springer-Verlag, London, UK, pp. 165–175.

Nierstrasz, O., Meijler, T.D., 1995. Research directions in software composition. ACM Comput. Surv. 27, 262–264.

Perry, D.E., Porter, A.A., Votta, L.G., 2000. Empirical studies of software engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering, ICSE '00. ACM, New York, NY, USA, pp. 345–355. http://doi.acm.org/10.1145/336512.336586

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering. In: International Conference on Evaluation and Assessment in Software Engineering (EASE). University of Bari, Italy.

Schneider, J.G., Han, J., 2004. Components: the past, the present, and the future. In: International Workshop on Component-Oriented Programming (WCOP).

Shaw, M., 2003. Writing good software engineering research papers: Minitutorial. Proceedings of the 25th International Conference on Software Engineering, ICSE '03. IEEE Computer Society, Washington, DC, USA, pp. 726–736. http://dl.acm.org/citation.cfm?id=776816.776925

Shull, F., Singer, J., Sjøberg, D.I., 2007. Guide to Advanced Empirical Software Engineering. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Sjoberg, D.I.K., Hannay, J.E., Hansen, O., Kampenes, V.B., Karahasanovic, A., Liborg, N.K., Rekdal, A.C., 2005. A survey of controlled experiments in software engineering. IEEE Trans. Softw. Eng. 31, 733–753.

Szyperski, C., 2002. Component Software: Beyond Object-Oriented Programming, 2nd Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2005. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requir. Eng. 11, 102–107.

Tassio Ferreira Vale is an assistant professor at Federal University of Recôncavo da Bahia, a graduate of computer science from Universidade Salvador (UNIFACS) in 2009, received his M.Sc. degree in Computer Science from Federal University of Pernambuco in 2012, and is currently a PhD candidate in Computer Science at Federal University of Bahia. He is an IT consultant, lecturer and software development freelancer. In addition, he is member of the Reuse in Software Engineering Research (RiSE) group and Fraunhofer Project Center (FPC) Brazil. His primary interest is software engineering, with an emphasis on software product lines traceability, component-based software engineering and software reuse.

Ivica Crnkovic is a professor of industrial software engineering at Mälardalen University where he is the scientific leader of the industrial software engineering research. His research interests include component-based software engineering, software architecture, software configuration management, software development environments and tools, and software engineering in general. Professor Crnkovic is the author of more than 150 refereed articles and papers on software engineering topics and a co-author and co-editor of two books: "Building reliable component-based Systems" and "Implementing and integrating Product Data Management and Software Configuration Management". He has co-organized several conferences and other events related to software engineering (such as ESEC/FSE, ASE, ECSA, Euromicro SEAA conferences, CompArch/WICSA federated conferences, etc.), and participated in Program Committees major software engineering conferences. His teaching activities cover several courses in the area of Software Engineering undergraduate and graduate courses.

Eduardo Santana de Almeida is an assistant professor at Federal University of Bahia and head of the Reuse in Software Engineering Labs. He has more than 100 papers published in the main conferences and journals related to software engineering, and has chaired several national and international conferences and workshops. His research interests include methods, processes, tools, and metrics to develop reusable software. Contact him at esa@dcc.ufba.br.

Paulo Anselmo da Mota Silveira Neto has a Bachelor of Computer Science degree from Catholic University of Pernambuco (UNICAP), specialist in software engineering from University of Pernambuco (UPE), Master of Science degree in computer science (software engineering) from Federal University of Pernambuco (UFPE). Nowadays, he is a PhD candidate in computer science at Federal University of Pernambuco and member of the RiSE group, which has executed research regarding to Software Product Lines (SPL) Testing, SPL Architecture Evaluation, Test Selection Techniques, and Regression Testing. He is also participating on important research projects in software engineering area, as the National Institute of Science and Technology for Software Engineering (I.N.E.S.).

Yguarat Cerqueira Cavalcanti is a graduate of computer science from Federal University of Alagoas in 2007, received his MS degree in Computer Science from Federal University of Pernambuco in 2009, and PhD in Computer Science at Federal University of Pernambuco. He is a system development analyst at the Brazilian Federal Organization for Data Processing (SERPRO). He is member of the Reuse in Software Engineering Research (RiSE) group, ACM, and Institute of Electrical and Electronics Engineers Computer Society. His primary interest is software engineering, with an emphasis on software maintenance and reuse.

Silvio Romero de Lemos Meira has a degree in electronic engineering from the Instituto Tecnológico de Aeronáutica (1977), Masters in Computer Science from Universidade Federal de Pernambuco (1981), and PhD in computer science at University of Kent at Canterbury (1985). He is currently a professor at Universidade Federal de Pernambuco. He has experience in computer science, with emphasis on software engineering, working on the following topics: software reuse, information systems, open source, social networking, performance, and quality metrics in software engineering.