UNIVERSITY OF GOTHENBURG

Status of Empirical Research in Component Based Software Engineering

A Systematic Literature Review of empirical studies

Master of Science Thesis in Software Engineering and Management

BHARATH TEKUMALLA

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, January 2012

Status of Empirical Research in Component Based Software Engineering
A Systematic Literature Review of the empirical studies

BHARATH TEKUMALLA

Examiner: ROBERT FELDT
Supervisor: SVEN-ARNE ANDRÉASSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden January 2012

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my supervisor Dr. Sven-Arne Andréasson for his continuous support and guidance throughout my work.

Finally I'm grateful to my family and friends who were always there to attend my needs and I'm happy for their care, best wishes and blessings.

Abstract

Objective: In this paper we present a systematic literature review of the empirical research in Component Based Software Engineering (CBSE). CBSE has evolved as a popular software development methodology since the introduction of Microsoft's Component Object Model (COM) in the early 90s. The purpose of CBSE is to develop systems by incorporating various independent yet well-defined software pieces in the name of components. The objective of this study is to identify the amount of empirical research done, the types of empirical studies and the research topics that are being discussed in the literature.

Method: We performed a systematic literature review of the papers that were published between January 1995 and August 2011. CBSE attained much of the industry's attention only after the introduction of Microsoft's COM, Sun Microsystems's JavaBeans and OMG's CORBA in the early 90s which showed up after 1993, thus we chose 1995 as the starting point for research on CBSE. We followed the guidelines of Kitchenham in performing the review.

Results: We found 47 papers which is the amount of empirical research that has been done during this period. Case study research and Experimentation were the most prevalent and preferred research methodologies which constituted 40.5% and 42.5% respectively. The research topics that were the most discussed among these papers are *Implementation of Components*, *Selection of Components* and *Quality of Components* which constituted 14.9%, 12.8% and 10.6% respectively.

Conclusion: From this study we found certain areas of CBSE (*Integration*, *Testing* and *Storage of Components*) which we consider necessary to be researched through Industrial Case Studies and Experiments as valuable insights of the current-state-of-practice in the industry can be explored. Regarding the industrial empirical research we observed that much of the studies were done in Europe where we highlight the need for a more geographical prevalence of industrial research considering the benefits of a socio-economic and business environment. Finally, we identified few interesting topics or subjects regarding the CBSE process which were not focused in the empirical research that has been done so far.

Table of Contents

1. Introduction 7
2. Background 8
   2.1 Software Components 8
   2.2 Component Based Software Engineering process 10
3. Method 13
   3.1 Research questions 13
   3.2 Search process 14
   3.3 Study selection 15
   3.4 Process followed 15
   3.5 Quality assessment 16
   3.6 Data collection and analysis 18
4. Results 18
   4.1 RQ1 – How much empirical research has been done 18
   4.2 RQ2 – Types of empirical studies 20
   4.3 RQ3 – Research topics being addressed 21
5. Discussion 23
   5.1 Case Studies 23
   5.2 Experiments 28
   5.3 What's missing? 32
6. Current state of CBSE 32
   6.1 Implementation of Components 33
   6.2 Selection of components 34
   6.3 Quality of Components 36
   6.4 Reusability of components 37
   6.5 CBSD Process 39
   6.6 Performance of components 40
   6.7 Component Testing 41
   6.8 Storage of components 42
   6.9 Integration of components 43
   6.10 Implementation and Maintenance of components 43
   6.11 Design and Implementation of components 44
   6.12 Component Architecture 44
   6.13 Maintenance of components 45
   6.14 Extensibility of components 45

7. Limitations of this study 46
7.1 Completeness 46
7.2 Data synthesis 46
7.3 Potential bias 46
8. Future work 47
9. Conclusion 47
10. References 48

1. Introduction

In 1980 object oriented technology came into existence and that enabled software reuse in a broader scope including the reuse of class analysis, design and implementation [1]. Many object oriented C++ class libraries were developed as reusable software packages. Thus object oriented technology steered the evolution of component technology from reusable functional libraries to object class libraries. In 1990 many large corporations (IBM, HP, Lucent Technologies) launched enterprise oriented software reuse project to develop domain specific business components for product lines using object oriented technology [1]. At this point of time the Object Management Group (OMG) began to standardize an open middleware specification for distributed middleware application systems and developed Common-Object Request Broker Architecture (CORBA). The object management group also specified a set of CORBA object services that defined standard interfaces to access common distribution services, such as naming transactions and event notification and all these are done to provide a high level reusable components [1].

Component Based Software Development or Engineering (hereafter we use CBSD and CBSE interchangeably) has evolved as a popular software development methodology since the introduction of Microsoft's Component Object Model (COM) in the early 90s. CBSD is claimed to be a process that produces software of high quality and also a process that reduces the product's time-to-market, which are the characteristics that are considered by the industry to be vital for a software product. In this development methodology major emphasis is put on disintegration of the designed systems into practical and logical reusable components. Apart from the characteristics mentioned just before, reusability and reusable components are also the vital aspects which stand as the backbone for the CBSD process. There has been an abundant amount of research done on various aspects, phases and characteristics of the CBSD process since its inception in the industry.

To this end, we wanted to review the literature that has been published on empirical research of CBSD since 1995 through 2011. We were interested to explore the state of empirical research on CBSD to see that if there are any areas of CBSD that are yet to be touched in the research process. The reason for focusing particularly on the 'empirical' studies is that empirical studies are the proofs of the hypothesis such as the one that we mentioned just before, about the industry's perception of CBSD process.

Prior to our study we read some of the literature in Software Engineering to get a complete understanding of how a literature review is to be performed and finally followed Kitchenham's guidelines for conducting a systematic literature review [2].

A significant work and the one that we mostly followed in our study was done by [3] which is a literature review of empirical studies conducted in Software Engineering that

were published in the journal – Empirical Software Engineering. Other works that motivated us were done by [4] and [5] as both the studies are purely based on the guidelines proposed by Kitchenham.

The rest of the paper is structured as follows: section 2 provides the background of this study which covers an overview of the process CBSD, section 3 explains the method we followed for this study at length, section 4 presents the results of our study, section 5 presents a discussion of the results, section 6 presents a summarizing picture of the current state of knowledge of CBSE, section 7 presents the limitations of this study, section 8 presents the future work that could be done which will finally be followed by the conclusion in section 9.

2. Background

In this section we present a general explanation of the CBSD process and its characteristics. This covers the description of Software Components and different phases of the CBSD process.

2.1 Software Components

The engineering practice of developing systems out of integrating individual parts that have independently been standardized and defined has been with us for some time now. This in fact dates back to the mechanization era and also the days of Henry Ford [6]. There are many pros associated with this form of engineering, among them including; marketing takes a short time, the cost and time associated with maintaining these systems is considerably low and most importantly these pieces can be reused across different products [7]. CBSD finds its inspiration from the success achieved by this engineering approach and with the aim of applying this engineering practice, the component based software development is adapted to develop systems by incorporating various independent yet well-defined software pieces in the name of components.

There is still some ambiguity when trying to virtualize the concept of components in software engineering, whereas in the other engineering disciplines the various components are touchable and therefore physical in sense and easier to grasp the concept of components, in software engineering this is not clearly defined. It is evident that the practice of components is really popular given the number of definitions one can find. There are at least fifteen definitions trying to give meaning to components but out of all we chose the definition provided by [8] as it gives out a small yet comprehensive picture of a software component. According to [8] the concept of components can best be approached from the perspective of its fundamental characteristics in order to fully understand it. His definition goes as follows –

> *"A software component is a unit of composition with contractually specified interfaces and context dependencies only. A software component can be deployed independently and is subject to composition by third parties".*

This definition also highlights the major properties of software components that are not addressed in traditional software modules. The most important characteristics of any component are its interfaces.

The interface specifies the entry point or the access point, to the functions of the particular component. These functions in most cases comprise all the operations contained in a component. However there is a distinction between two particular interfaces; a required and provided interface. Whenever a component makes a request for functionality for the purpose of accurate operation this interface is called as required interface. On the other hand, whenever this component is describing its own functionality this interface is called as a provided interface. From this we can see that the purpose of an interface is to enable a component to interact with other components and with that of the external environment which furthermore helps link these components together.

Despite the fundamental character of component concept and interface, there is a notion of component model. The component model provides a benchmark for which all the properties and restraints of the component and their manipulation tools must fulfill. The main concern for component model is towards the provision of component characteristics specification rules and components composition mechanics and rules with inclusion of properties. From this perspective it is therefore proper to say that the standardization keystone for software development is defined in the component model.

Looking at what [7] view components as; they define them on the basis of component model –

> *"A software component is a software facet that is conventional to the component model and can be independently setup and composed with no amendment according to a composition custom".*

The component based software development process is performed based on a well-defined component model with consistent component standards and approaches to support component interactions, customization, packaging and deployment. Before beginning of the development process, component domain analysis and modeling is performed first in order to come out with a domain-specific business model to support the definition of the components requirements.

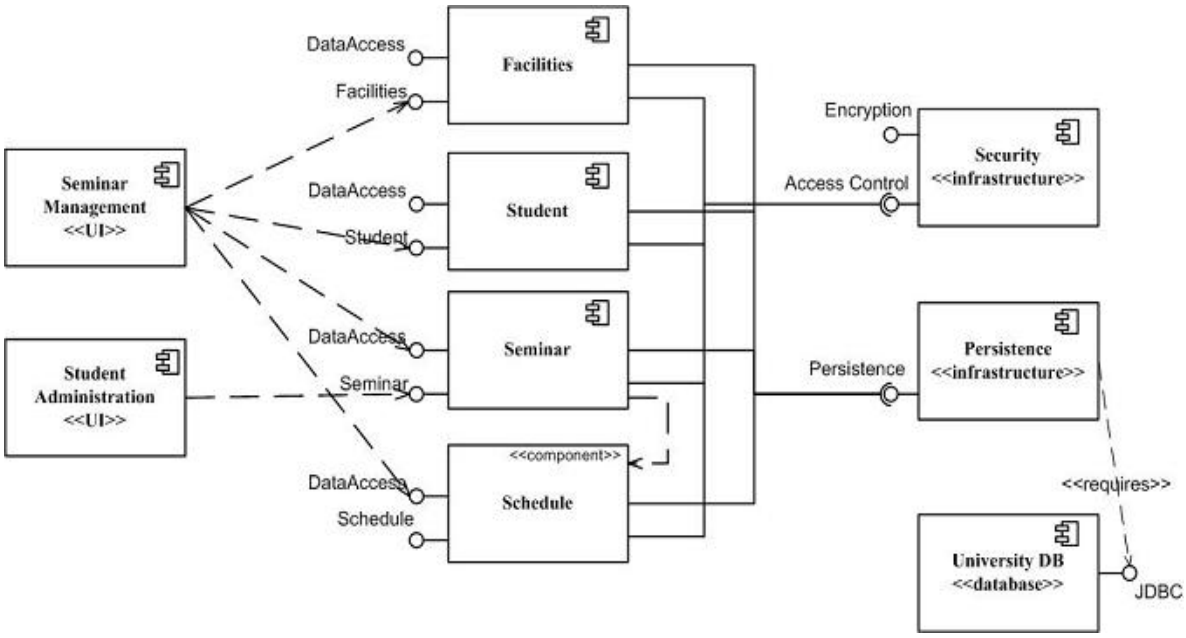

Figure 1 Component diagram in UML

Component oriented UML is used to define components by specifying component use-cases object-oriented structure and dynamic behaviors. Figure 1 shows an example of a university's administrative system developed in UML's component diagram. This example is shown to give an understanding about the approach for domain modeling and analysis. UML has been the most preferred way for modeling since the beginning of the Object Oriented era. As we mentioned before that objected oriented technology has paved the way for component oriented concept, the component oriented technologies inherit the characteristics of object oriented technologies such as this example of domain modeling and the use of UML for it.

2.2 Component Based Software Engineering process

The life cycle of component-based software development flow seems to be a standard waterfall development process but bear in mind that increments and iteration will occur. Incremental development is a technique for identifying priorities and delivering high priority items first. Iteration is a technique where a basic infrastructure is built upon, here the infrastructure maps to software development as early versions of software evolve over time. Figure 2 taken from [9] provides a clear picture of the CBSD when incorporated into the traditional waterfall process.

[9] also presents a very comprehensive description of CBSD in regard to its various phases of development. We present a summarizing description of their work with respect to the phases of the process as follows –

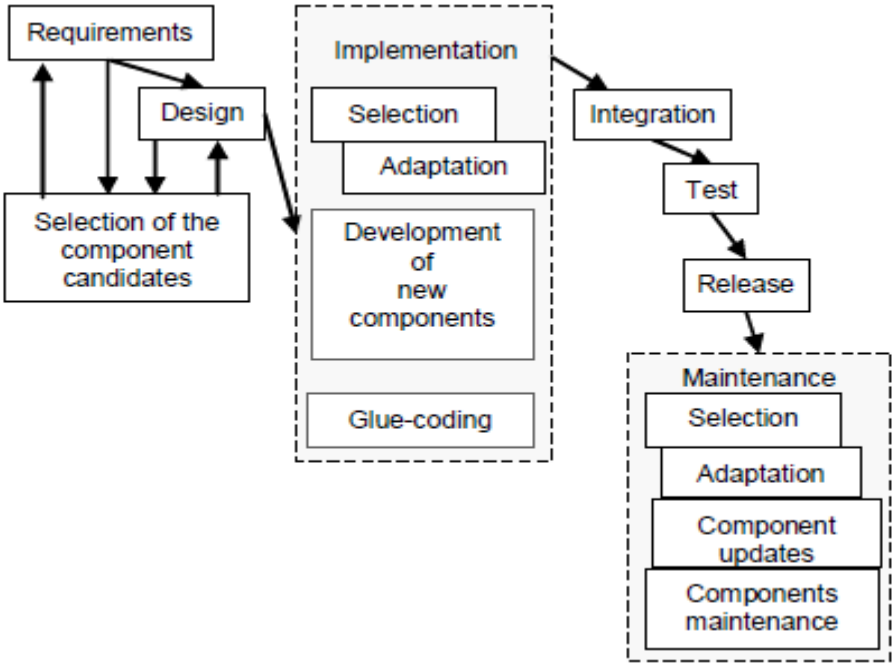

Figure 2 Component based Waterfall product lifecycle [9]

The component-based software engineering process consists of the following six phases:

- Requirements Analysis
- Design
- Component identification and customization
- System Integration
- System Testing
- Software Maintenance

Requirements Analysis:

In this phase all the component requirements like functional and non-functional are collected, analyzed and specified based on a well-defined methodology such as UML. The result of this phase is a component specification document.

Design:

In this phase engineers design components based on the component requirements specification from the previous phase. The component design includes three tasks. The first task is to conduct component design for functional logic and data objects and make trade-off decisions on technologies and operation environments. The second task is to follow a selected component model and work on component realization by providing data exchange mechanisms for component communication and interactions. The final task is to define consistent approaches to support component packaging and deployment. The outcome of this phase is the Design Specification Document.

Component identification and customization (Coding):

As we mentioned in the earlier sections that reusability and reusable components form the backbone of the CBSD process, in this phase suitable components are identified and are customized apart from specifying new components which is done in the previous phases. Implementation of the components is performed using a specific technology and programming language based on the design and targeted operating environments. The focus is on composing and assembling components that are likely to have been developed separately, and even independently. Component identification, customization and integration are the crucial activities in the life cycle of component-based systems. It includes two main parts:

- Evaluation of each candidate component, based on the functional and quality requirements that will be used to assess that component.
- Customization of those candidate components which should be modified before being integrated into new component-based software systems.

System- Integration:

It is possible for a component to be implemented for more than one operating environment. Each implemented component depends on a specific technology set and a targeted operating environment. Each component that is identified and customized are integrated together to meet the specifications. Integration is to make key decisions on how to provide communication and coordination among various components of a target software system.

System-Testing:

In this phase the component is validated based on a given specification and design. During this phase, component testers perform software testing such as white-box and black-box testing to uncover various errors. Since software components are delivered as

a final product, component testing plays an important role in the process and it includes component usage testing, performance testing and deployment testing.

Software Maintenance:

This phase begins after shipping the first version of a software component or the complete product to the customer. In this phase, Software components are updated and enhanced to meet customer requests and to resolve discovered problems.

3. Method

In this section we describe the method we followed for our study in detail. This covers the research questions of our study, the search process we followed in order to accumulate the relevant literature, the selection criteria of the literature, quality assessment of the selected literature and finally the procedure we followed for extracting the data from the literature.

3.1 Research questions

The research questions for our study are

RQ1). How much empirical research has been done in CBSE since 1995?

RQ2). What types of empirical research has been done on CBSE?

RQ3). What research topics are being addressed by these empirical studies?

Regarding RQ1, it may be a concern for choosing the year 1995 to be as a starting point for gathering the papers. The reason for this is that, we understood from [7] and [8] that the evolution of CBSE/CBSD shows that since its inception in the form of Object Oriented technology, much of the attention attained to it was only since the introduction of component technologies like Microsoft's COM, Sun Microsystems's JavaBeans and OMG's CORBA in the early 90s. Hence we decided to start our search for papers that were published after their introduction which shows up after 1993, thus we started at 1995.

Regarding RQ2, we were interested in knowing the types of empirical studies that were conducted with respect to CBSD or CBSE or Component Based Software (CBS). We adapted the classification of studies from [3] which is shown in Table 1. The table shows different types of studies that were conducted in the field of software engineering over all these years. For example, if a study consists of two methods such as a case study and an experiment that follows it, then we consider the study as a case study. Similarly if a study consists of literature review which is succeeded by other methods, then we consider the study to be the latter method followed.

Table 1 Type of studies [3]

Study	Definition
Case study	In-depth analysis of a particular project, event, organization, etc.
Correlational study	Measuring variables and determining the degree of relationship that exists between them.
Observational study	Observe, record and analyzing the results without the investigator's intervention into the setting.
Experiment	Quantitative study to test cause-and-effect relationships.
Survey	Data is collected by interviewing a representative sample of some population.

With respect to RQ3, we wanted to explore the areas concerning CBSD or CBSE or CBS that were covered by the empirical studies, for instance, the number of studies that were published on the issue of quality of the components.

3.2 Search process

We derived certain keywords from the research questions and we used them to extract the papers from different databases. The keywords were –

CBSD, Software development, organization, CBSE, component, component based software, empirical study, empirical research

Synonyms for the keywords –

Company* OR Corporation* OR Inc.* AND empirical study OR survey OR case study OR experimentation OR empirical research AND software component OR component life cycle OR software development process OR component development

The databases we used for our search are SpringerLink, IEEE Xplore, ACM Digital Library and Elsevier. These are the databases that contain much of the literature related to Software Engineering that was published by various international conferences, journals and notes by Special Interest Groups, SIGSOFT for instance.

3.3 Study selection

In this section we present in what follows are the criteria which we followed in including and excluding the studies for our study.

Inclusion criteria

- Papers that were published from January 1995 to August 2011 were selected
- Papers that state their study as an empirical study in their titles or has at least one type of empirical study as part of their study, for example, a study which has a literature review as its primary study and validates the results obtained from the literature review through an empirical study, were selected for our study.
- Studies that focus on aspects of CBSD, Component Based Software (CBS) or CBSE were selected
- Papers containing the keywords 'empirical', 'survey', 'case study' or 'experiment' relating to CBSD, CBS or CBSE were selected

Exclusion criteria

- Opinion papers, "lessons learnt", view point or position papers and studies that are not empirical or does not contain an empirical study as part of their whole study were excluded
- Papers that are external to CBSE or CBSD or CBS were excluded, for example, a paper discussing about reusability but not in regard to software components.
- Duplicate reports of the same study were excluded

3.4 Process followed

Figure 3 shows an illustration of the process we followed in collecting the studies. We initially searched for papers on CBSE without including the keyword 'empirical' or any of its synonyms in the search string. This is in order to get the total number of studies on

CBSE which includes literature reviews, opinion papers, position papers, etc. from the 4 databases which resulted in 127,282 papers.

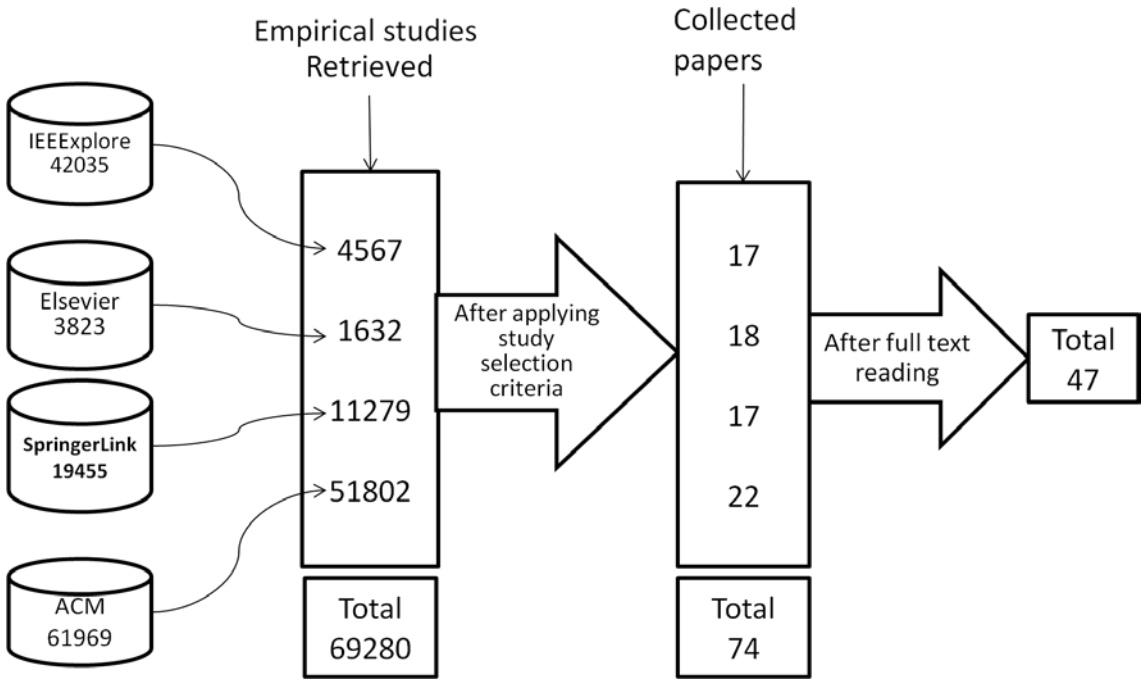

Figure 3 Selection of studies

Later we included the keyword ‘empirical’ and the other synonyms in the search and it resulted in 69280 papers which contained studies that had the keyword ‘empirical’ in their title, list of keywords or abstract. This number is shown in the figure as total retrieved empirical studies and also the number of papers that were retrieved from each of the database is also shown.

After applying the study selection criteria, which we explained in section 3.3, we were left with a total of 74 studies which is shown in the figure. These included duplicate studies, studies that contained the keyword ‘empirical’ and its synonyms in their titles, list of keywords or abstracts. After doing a full text reading of these studies we removed some of the studies that were duplicate, contained some of the search keywords but the studies were actually not related to CBSE or CBSE and position papers and finally ended up with 47 studies.

3.5 Quality assessment

We framed a set of quality assessment criteria which we extracted from [2] and modified so as to make them adaptable for our study. Following is the criteria which is based on four questions –

QA1 – Is the research question (or hypothesis) stated clearly in the study?

QA2 – Is the type of research method (experiment, case study, survey) clearly explained?

QA3 – Were the findings and analysis clearly presented?

QA4 – Are the authors' claims about the conclusions justified by the data?

With QA1 we assessed whether the objective or aim of the study was clearly explained either in the introduction or background sections of the paper. This includes the presentation of the research questions or hypothesis that would be addressed by the study. We evaluated the studies by reading and interpreting these two sections and mainly depended on the text that implicitly or explicitly expressed the intention or objective of the study.

With QA2 we assessed whether the method followed for the study was clearly explained in terms of its repeatability and other parameters. The parameters we considered mainly were:

- The type of study that would be followed – whether it would be a qualitative or quantitative study
- Description of the study setting or context – whether it takes place in an industrial setting or in an academic environment
- Description of the subjects involved in the study – whether humans or systems were the subjects
- Description of the sample chosen if humans were the subjects – what type of sampling strategy was considered for e.g. random sampling or purposive sampling, etc.

With QA3 we assessed whether the results obtained from the study were clearly presented in terms of sensibility and appropriateness. This means that if the study was stated to be a qualitative one, then we considered the way the results were presented i.e. whether they were explanatory or not. If the study was stated to be a quantitative one, then we looked out for the results whether they were numerically or statistically presented or not. Regarding the sensibility of the results, we checked whether the results were answering the research questions stated for the study or validate the hypothesis stated for the study by reading and interpreting the results. For papers with results of hard-to-understand type, we read and interpreted only certain parts of those results and assessed the quality, for example, papers with mathematical results or results with heavy usage of various mathematical symbols, formulae, etc.

With QA4 we assessed whether the conclusions drawn from the study had a mapping to the data presented in the results section or not. We evaluated this by reading the conclusion section of all the studies where we mainly depended on the text that sounded as a summary to the whole study referring to the data that has been dealt in the work.

3.6 Data collection and analysis

We followed a classification scheme as was followed by [3] in their study. The classification is done based on the research topic being addressed, the method followed and the source of data. Following are the steps we followed:

- Reading the abstract and conclusion or summary
- Writing down the issue or topic that the study presents
- Identifying the area associated with CBSD or CBSE or CBS that the issue discusses
- Mapping the area associated with CBSD or CBSE or CBS to a traditional development process like Waterfall process. For example if the paper discusses about the subject of integration of components it would be mapped to the implementation phase of Waterfall process.
- The type of empirical study performed.

4. Results

In this section we present the results of the study.

4.1 RQ1 – How much empirical research has been done

Forty-seven papers were identified as answering the question "*How much empirical research has been done with respect to CBSD or CBSE or CBS since 1995?*"

Our search with the keywords fetched around 69,280 papers from the four databases that we mentioned in section 3.2. After applying the document selection criteria we extracted only 74 papers, where the rest of the papers were mere instances with occurrences of the keywords within them. When we started with the activity of data retrieval from these papers, we noticed that 27 out of these 74 papers were literature reviews (25) and position papers (2) and therefore we removed them. Thus the final list consisted of 47 papers which present the gross figure for the amount of empirical research done since 1995.

We present the quality assessment of all the accepted papers in table 2. We used a 3-point scale which consists of the points 'Yes', 'Partially' and 'No' to denote the satisfactoriness of each of the criterion. If a study's objective or aim was clearly described then it would be rated with a 'Y' representing the point 'Yes'. If the method followed was not clear enough to be understood, then it would be rated with a 'P' representing the point 'Partially' and similarly with 'N' representing the point 'No' if the description was not clear at all.

An example for grading a criterion as 'Y' is, the study [10] where it clearly specifies in its introduction that the goal of this paper is to find a suitable heuristic to minimize change during component system evolution through CDR. It was clear for us from the text that the aim was to find something new from the study, therefore we graded the criterion QA1 with a 'Y' in this case.

An example for grading a criterion as 'P' is, the study [11] in which it was stated that an empirical study would be conducted to propose a security mechanism for CBS called CASSIA (explained in section 6.6). However, it was not clear about the design, settings and how the study was executed. Merely a hypothesis and the context of the study were explained. Therefore we graded the criterion QA2 with a 'P' in this case.

Finally for grading a criterion as 'N' we present two example studies. In the study [12], it was just stated that an experiment was conducted on some components to evaluate a metric that was proposed in the same study and no further explanation about the design, execution of the study and context was available. Moreover the results obtained from the study were just presented in tables and some description was provided which we found to be very difficult to understand. Therefore we graded the criterion QA2 and QA3 with 'N' in this case. Similar is the case with [13] in which it was stated that an experiment was conducted on a component to evaluate its quality based on a quality model that has been proposed in the same study. However it lacks with explanation about the design, execution and context of the study and also we found the results very difficult to interpret. Therefore we graded QA2 and QA3 with 'N'.

Table 2 Quality assessment of the accepted papers

	Description	Yes (Y)	Partially (P)	No (N)
QA1	Is the research question (or hypothesis) stated clearly in the study?	45 (~95.7%)	2 (~4.3%)	0
QA2	Is the type of research method clearly explained?	36 (~76.6%)	9 (~19.1%)	2 (~4.3%)

QA3	Were the findings and analysis clearly presented?	37 (~78.7%)	8 (~17.0%)	2 (~4.3%)
QA4	Are the authors' claims about the conclusions justified by the data?	40 (~85.1%)	7 (~14.9%)	0

4.2 RQ2 – Types of empirical studies

We present the result for the research question RQ2 i.e. "*What types of empirical studies have been done with respect to CBSD or CBSE or CBS?*" in Figure 4.

Figure 4 shows 5 types of studies that are prevalent in the research on CBSD or CBSE or CBS. The most preferred methodologies are Experiments and Case studies and they constitute 42.5% and 40.5% of the total number of studies (47) respectively. Surveys constitute 10.6%, Correlational studies constitute 4.3% and Observational studies constitute 2.1% of the total number of studies.

Most of the experimental studies were conducted as part of their major studies where 12 out of the total 20 experiments employed humans as the subjects comprising students. The remaining 8 were on technical aspects and in this case they were on Software Components.

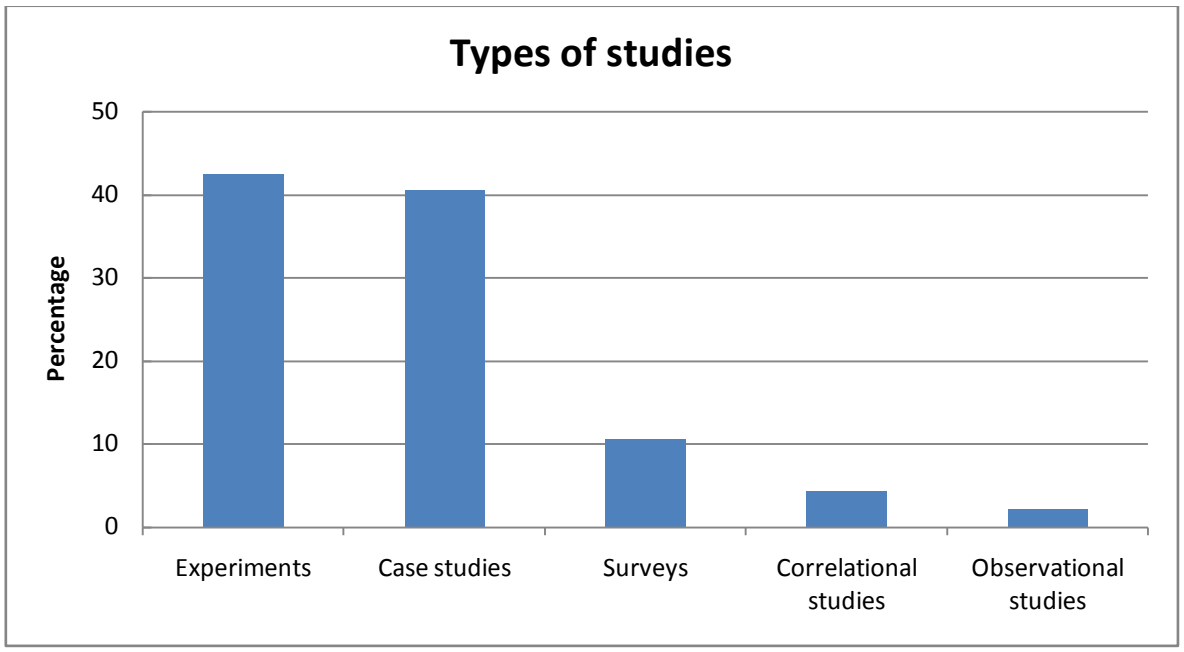

Figure 4 Types of empirical studies

The case studies resulted in three types – 10 industrial, 6 qualitative and 3 quantitative case studies. Industrial case studies are those which were conducted in an industrial setting, for instance, conducting interviews with the employees of a company or using the documentation associated with a system used in a company for analysis in the study. Furthermore, the industrial case studies were divided as two types – qualitative and quantitative. There were 7 qualitative and 3 quantitative industrial case studies. The difference between them lies in their methodology followed.

Regarding the amount of the remaining empirical studies, the number of surveys conducted were 5, Correlational studies were 2 and 1 observational study.

4.3 RQ3 – Research topics being addressed

We present the result for the research question RQ3 i.e. "*What research topics are being addressed by the empirical studies?*" in Figure 4.

Figure 5 shows all the research topics that are being addressed by the empirical studies. The most discussed topics are *Selection of Components*, *Implementation of Components* and *Quality of Components*.

Implementation of Components constitute 14.9%, *Selection of Components* constitute 12.8% and *Quality of Components* constitute 10.6% of the total number of studies (47). The next most discussed topics are *Reusability of Components* (10.6%), *CBSD Process* (8.5%), *Performance of Components* (6.4%) and *Component Testing* (6.4%).

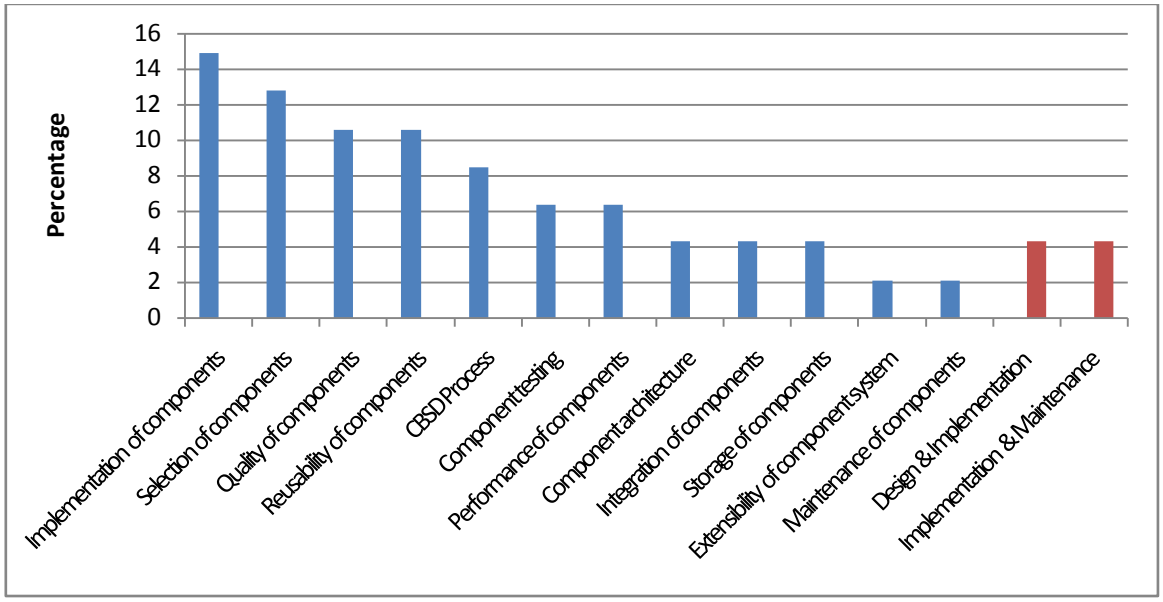

Figure 5 Research topics being addressed

The two bars with different color towards the right side of the figure represent the topics *Design & Implementation* and *Implementation & Maintenance*. These two constitute 4.3% each, of the total studies. The reason for showing them in different color is that these topics were discussed jointly in the studies. For instance, the topics design and implementation of components were both discussed in a single study therefore we treated both of them together as a category in itself without splitting them into two. Furthermore, there were 3 sub-topics *viz. Configurability*, *Changeability* and *Tailorability* that were discussed under the category *Implementation of Components*.

Table 3 presents a frequency of the studies that were discussing all the above topics.

Table 3 Frequency of research topics

Topic	Studies	Frequency
Implementation of components	[14] [15] [16] [17] [18] [19] [20]	7
Selection of components	[21] [22] [23] [24] [25] [26]	6
Quality of components	[27] [28] [29] [12] [13]	5
Reusability of components	[30] [31] [32] [33] [34]	5
CBSD Process	[35] [36] [37] [38]	4
Performance of components	[11] [39] [40]	3
Component testing	[41] [42] [43]	3
Storage of components	[44] [45]	2
Integration of components	[46] [47]	2
Implementation and	[48] [49]	2

Maintenance		
Design and Implementation	[50] [51]	2
Component architecture	[52] [53]	2
Maintenance of components	[54]	1
Extensibility of component systems	[10]	1

5. Discussion

In this section we present our discussion based on the answers that we presented in the previous section. We discuss about the three research methodologies that were most prevalent in the empirical research of CBSD with respect to the topics that were researched in the studies. We first present about case studies which will be followed by our analysis on experiments.

5.1 Case Studies

Case study research has been quite prevalent in the field of Software Engineering through all these years. According to [55] Case Study is a research methodology which can either be qualitative or quantitative in nature with different characteristics such as exploratory, explanatory, descriptive or improving. The method followed in a case study depends on the setting or context in which the study takes place, industrial case study for instance, that takes place in a company or organization. We present in what follows is the analysis of the data that we gathered in this study.

We presented the total number of case studies in section 4.2 which was 19 out of which the number of industrial case studies were 10. Table 4 presents the list of all case studies that were conducted in the industry. The table presents the description or a summary of the study and the area of the CBSD that has been the subject of the corresponding study.

Table 4 Industrial case studies

Paper ID	Description	Researched area of CBSD
P2 [48]	Study of demands on development and maintenance of	Implementation and Maintenance of reusable

	reusable components	components
P3 [35]	Human, social and organisational issues affecting the introduction of Component-Based Development (CBD) in organizations are presented	CBSD Process
P6 [49]	The issues and challenges encountered when developing and using an evolving component-based software system are discussed by doing a case study	Implementation and Maintenance of components
P14 [33]	Hypotheses about impact of reuse on defect density and stability and impact of component size on defects and defect density in the context of reuse are assessed.	Reusability of components
P18 [52]	The advantages and liabilities the use of a component-based software architecture entails for the development of an industrial control system are presented through an industrial case study	Usage of component-based software architecture
P22 [16]	Tailorability should enable users to fit computer systems to the application context. So tailoring options should be meaningful for end-users in their respective domains. This paper discusses how these design criteria can be realized within the technical framework of component-based tailorability.	Implementation of components (tailorability)
P24 [37]	Although previous studies have proposed specific COTS-based development processes, there are few empirical studies that investigate how to use and customize COTS-based development processes for different project contexts. This paper describes an exploratory study of state-of-the-practice of COTS-based development processes.	CBSD Process
P35 [18]	To see whether the benefits associated with TDD can be shown for reusable components	Implementation of components

P36 [19]	Development with OSS components faces challenges with respect to component selection, component integration, licensing compliance, and system maintenance. Although these issues have been investigated in the industry in other countries, few similar studies have been performed in China.	Implementation of components
P42 [26]	The actual industrial practice of component selection in order to provide an initial empirical basis that allows the reconciliation of research and industrial endeavours, is investigated	Selection of components

From the table it is apparent that much of the research, that has been done, had focused on the topics *Implementation*, *Design* and *Maintenance* of components. This gives us a perception that the other areas such as *Integration*, *Testing* and *Storage of components* which are also vital in the process of CBSD were less researched in an industrial setting. Regarding the research on *Integration of Components*, we observed that the number of industrial case studies is not adequate enough to cover various issues associated with it. Some of the issues as discussed in the literature were the design trade-offs in component integration, incompatibility between the components and interaction between them [56]. Therefore we would like to highlight that these issues can be brought under discussion through industrial case studies as the methodology offers flexibility in organizing the study and also the outcome of the studies would give a valuable insights of the industry.

Another point of observation we had is about the geographical distribution of these industrial case studies. Table 5 shows the list of countries in which the case studies took place.

Table 5 Geographical distribution of Industrial Case Studies

Paper ID	Country
P2 [48], P6 [49], P18 [52]	Sweden
P3 [35]	UK
P14 [33], P24 [37], P35 [18]	Norway

P22 [16]	Germany
P36 [19]	China
P42 [26]	Spain, Norway and Luxemburg

From the table we can clearly notice that Norway and Sweden has more case studies than other nations with 4 and 3 studies respectively. This shows a tendency that the IT industry of Norway and its neighbor Sweden have about the collaboration with academia in order to maintain a good technological, scientific and sustainable profile.

From a global perspective, much of the industrial research on CBSD has been done only in Europe with China as the only exception. From this we would like to generalize our view and highlight that it is necessary that other developed nations like US, Canada, Australia etc., and developing nations like India, China, etc., should also conduct industrial research as it would be very helpful for the IT industry on the global stage with such knowledge transfer mechanism from a socio-economic and business point of view.

Apart from the 10 industrial case studies, 8 were simple case studies which are presented in Table 6.

Table 6 Case studies

Paper ID	Description	Researched area of CBSD
P8 [50]	The best practices in designing and building a web-based auction system by using UML and components are presented through an empirical study	Design and implementation of components
P17 [15]	Comparison between Object-Oriented building and Aspect-Oriented building of components in regard to the changeability of the system	Implementation of components (Focus on changeability)
P25 [28]	Most previous works on software quality evaluation are focused on COTS-based software or deliverable software products with quality	Quality of components

	model and metrics. However, this paper has presented a quantitative quality evaluation approach with respect to the Component Based Development (CBD) methodology of Ministry of National Defense of Republic of Korea.	
P26 [11]	A scalable security mechanism named CASSIA, for component based systems is proposed	Scalability and performance of components
P27 [29]	Facilitating design decisions by making accurate predictions of how failure-prone a component will be – an empirical study on ECLIPSE Plugins	Quality of components
P28 [51]	Two critical aspects of component based systems in the financial industry are addressed - Component based design of systems and the mediation between the components	Design and implementation of components
P29 [38]	CBD will improve globally distributed software development practices by allowing each site to take ownership of particular components, resulting in reduced inter-site communication and coordination activities. Such an approach may indeed overcome breakdowns in inter-site coordination efforts; however, it may also lessen opportunities to share knowledge between sites and may hamper opportunities to reuse existing components. A case study approach, exploratory in nature, was adopted to explore knowledge aspects in global component-based software development.	CBSD Process
P33 [34]	Discusses the modularity offered by Aspect-Oriented Programming and its association with obliviousness and the trade-offs between modularity and obliviousness are presented with respect to reusable components.	Reusability of components
P40 [25]	Improving the selection of OSS components	Selection of components

From the table we perceive that there has been good amount of research done through case studies and these studies cover most of the areas or phases of CBSD process. 5 out of these 8 studies were qualitative and the remaining 3 were quantitative in nature.

5.2 Experiments

Experimentation is one of the most preferred research methodologies in the field of Software Engineering. According to [55], experimentation is a research methodology which is quantitative by nature with an explanatory characteristic.

We differentiated between a Case Study and an Experiment by primarily considering the text presented in the papers that clearly/explicitly mentioned or described whether that study is a case study or an experiment. However there were certain cases for e.g. in [14] where the study was stated to be a case study and the method followed was said to be experimental i.e. the studying of the *case*, was carried out in the form of experiments. Thus we considered even such studies as experiments as our focus was mainly on empirical studies that were conducted in regard to CBSD or CBSE.

Table 7 presents a list of all the 20 experiments that were conducted with respect to the process of CBSD. We evaluated the studies by full text reading and finally ended up with this list.

Table 7 Experiments in the empirical research of CBSD

Paper ID	Description	Researched area of CBSD	Subjects
P1 [30]	Evaluation of published software metrics that would measure the benefit of reuse of components	Reusability of components	Students
P7 [27]	Study of consumers' preferences and purchasing behavior of software components regarding to the quality attributes of the components	Quality attributes of software components	Students
P10 [32]	An active reuse repository system called *CodeBroker* is used to show that active repository systems promote reuse by motivating and enabling software developers to reuse components whose existence is not anticipated, and reducing the cost of reuse through the automation of the	Reusability	Students

	component location process.		
P11 [44]	A scheme for classifying and describing business components and the design of a knowledge-based repository for their storage and retrieval is proposed	Storage of components	Students
P12 [14]	Component software provides better productivity and configurability by assembling software from several components. This paper investigates system configurations on a component-based system and the side effects of the configurations.	Implementation of components	Components
P13 [45]	Different component indexing and retrieval methods were tested and found that full-text indexing and retrieval of software components is better than controlled vocabulary indexing and retrieval	Storage of components	Students
P15 [57]	A major challenge i.e. compositional reasoning about the system Quality of Service is addressed by proposing an empirical reasoning approach	Quality of Service of components	Components
P16 [21]	Proposal of metrics for measuring similarities between component interfaces based on interface refactoring	Selection of components	Students
P19 [22]	The rigorous specification of components is necessary to support their selection, adaptation, and integration in component-based software engineering techniques. The specification needs to include the functional and non-functional attributes. The non-functional part of the specification is particularly challenging, as these attributes are often described subjectively, such as Fast Performance or Low Memory. Here, the use of infinite value logic, fuzzy logic, to formally specify components is proposed	Specification and selection of components	Components
P20 [41]	This paper addresses the issue of usability testing in a component based software engineering environment, specifically measuring the usability	Testing of components	Students

	of different versions of a component in a more powerful manner than other, more holistic, usability methods. Three component-specific usability measures are presented: an objective performance measure, a perceived ease-of-use measure, and a satisfaction measure.		
P23 [16]	Describes a first empirical study comparing two defect detection techniques – code inspections and functional testing in the context of product line development of reusable components	Inspection and Testing of components	Students
P30 [12]	A metric called Component Complexity Metric is proposed which may be used to limit the complexity of the component	Quality of components	Components
P32 [13]	All the quality attributes may not be of prime importance for a component application, thus a new quality model is proposed with new characteristics which may be very relevant to the context of components	Quality of components	Components
P34 [39]	The actual effort required for developing a performance prediction model is addressed by proposing a component-based prediction model named Palladio	Performance of components	Students
P37 [24]	Although a multiplicity of COTS selection method have been proposed in literature, most developer still select COTS products using ad hoc methods. One of the main reason being, COTS selection method do not provide all or most of the required support and guidance required for carrying out the COTS selection process. Therefore this study is aimed to find out differences if any, between 3 selection methods and to determine the ability of each of the methods to provide adequate COTS selection support and guidance.	Selection of components	Students
P38 [42]	Proposal of a component testing approach and its	Component	Students &

	experimental evaluation for its efficiency	testing	Employees
P41 [54]	An experiment investigating component collaborations in the OSGi/Eclipse component model is presented. The aim of the experiment is to demonstrate the benefits of using a formal contract language.	Maintenance	Components
P43 [53]	Component-based software development needs to formalize a process of generation, evaluation and selection of Composite COTS-based Software Systems (CCSS), enabling software architects to make early decisions; the Azimut approach and its associated software tool were proposed to tackle this problem. This article presents an experimental study conducted to compare Azimut approach with a Systematized Ad-Hoc approach, regarding generated solutions quality, cost and effort.	Component architecture	Students
P44 [10]	Finding a heuristic to minimize change side-effects during component system evolution through a process called Component Dependency Resolution (CDR)	Extensibility of component systems	Components
P45 [40]	Relation between autonomy and qualities of the system is studied by proposing an approach for quantifying autonomy	Performance of components	Components

From the table it is immediately apparent that the subjects involved in the experiments are mostly students. This observation of ours is in-line with the observation made in the study by [3] and therefore we would like to share his perception that, some important experiments could be conducted with industry professionals which would improve the generalizability of the results like as it was done by [42].

Another observation from the list is that though the experiments covered most of the areas of CBSD with *Selection of Components* and *Quality of Components* being the most researched, we would like to highlight other areas as well such as *Integration of Components* and *Design of Components* which could be researched through experimentation. For instance, evaluating the efficiency of a tool that supports the process of integration or experimenting with the design of a component and its interaction with others which could be done through controlled experimentation.

5.3 What's missing?

Here we present our discussion on those topics that were missing in the empirical research on CBSE. Each of the missing topics may either be considered as a part of one of the areas that we identified or as another area of CBSE itself.

- We didn't find any empirical study in the literature that has been done either supporting or opposing the popular claim of CBSE or CBSD that is this engineering practice 'lowers development cost and the product's time to market'. Few studies like [15] used this claim as one of the motivating factors for their studies but never had shown any interesting observation in this regard. Therefore it appears to us that some empirical research on this topic would be very useful. Probably industrial case studies or surveys can reveal the actual picture on how CBSE or CBSD reduces development cost and product's time to market thereby justifying the claim.
- There is no empirical research that we could find on the subject of 'component models' like COM, CORBA, etc. Indeed these two are the only well known component models so far since their inception and no new model have been introduced until now. It appears to us that the industry is content with these existing ones as of now but we predict that shortly in the future there will be a necessity for new component models according to the needs then. Therefore we see that the empirical research either on the existing ones or for introducing new ones would be very helpful for the industry.
- The area *Implementation of Components* lacks research on the tools that support the CBSE or CBSD process. We mean that there is no development environment or CASE tool available that is specifically designed to facilitate this process. For example, IBM's Rhapsody is a tool that is used to generate code from different UML models, which is an important characteristic of the latest software development practice Model Driven Development. Similarly it would be helpful if such a tool that can support the CBSE process be available and we believe that this is possible only through empirical research that can focus on the current industrial practices of CBSE which might lead to interesting ideas in this direction.

6. Current state of CBSE

In this section we present the current state of knowledge of CBSE with respect to all the areas of CBSE that we identified in the literature. We summarize the work done in each empirical study that has been done in each of the areas. We present our findings of each area according to the order presented in Table 3 in section 4.3.

6.1 Implementation of Components

This area of CBSE is the most researched through empirical studies which constituted 14.9% of the total number of empirical studies done. The focus of research was on the development of components in regard to different factors. The following descriptions are a summary of all the studies that were done in regard to the implementation of components.

- In the study [14], the side effects of the system configurations on a component based system are investigated. A component based java virtual machine has been implemented by modifying an existing virtual machine for the study. Some problems have been identified in order to use the system after such configuration as the study pointed out the dependencies among the components as the problem which needed a clear understanding of their behaviours.

- The study [15] is about investigating the claims of Aspect-Oriented Programming (AOP) in the context of COTS based systems. According to [15] the claims are such that AOP makes it easier to reason about, develop and maintain certain kinds of application code. In order to investigate these claims a case study was performed by comparing object oriented version and aspect oriented version of an application with respect to its changeability. Results of the study showed that heterogeneous glue code does not bring benefits in the context of AOP and to integrate COTS components using AOP, the tools that support this process are needed to be investigated.

- [16] is a study which states that tailorability should enable users to fit computer systems to the application context. So tailoring options should be meaningful for end-users in their respective domains. Thus a case study has been done to show how these design criteria can be realized within the technical framework of component-based tailorability. The study shows that a specific preparatory activity is required before following any tailoring activities. It also shows that a domain-specific requirement analysis of tailoring needs is necessary to solve the trade-off tailorability and complexity of the system.

- The study presented in [17] is about the effects of Open Source Software (OSS) components reuse on the development economics with respect to cost, productivity and quality. The study states that organizations can benefit from reusing OSS components in terms of productivity and product quality if they implement the components reuse adoption in a systematic way. The study was a qualitative exploratory study involving interviews of industry professionals. It has been stated in the study that a lesson learned during the study was that OSS components are of highest quality provided the company follows good practices when implementing them.

- The study presented in [18] is about investigating the benefits of Test Driven Development (TDD) in using reusable components. The study investigated defect and change density in relation to the use of TDD vs. test-last approaches on a framework of reusable components. The study finally discusses both benefits and drawbacks of using TDD. The results of the study showed that the relative change in mean defect and change density as 35.86% and 76.19% respectively.

- In the study [19], a survey has been done in China to investigate the challenges associated with software development with OSS components with respect to component selection, integration, licensing compliance and system maintenance. The results were stated as follows:

    - The main motivation behind using OSS components was their modifiability and low license cost. Using a web search engine was the most common method of locating OSS components.
    - Local acquaintance and compliance requirements were the major decisive factors in choosing a suitable component.
    - To avoid legal exposure, the common strategy was to use components without licensing constraints.
    - The major cost of OSS-based projects was the cost to learn and understand OSS components. Almost 84% of the components needed bug fixing or other changes to the code. However, close participation with the OSS community was rare.

- According to [20], software is often built from pre-existing, reusable components, but there is a lack of knowledge regarding how efficient this is in practice. Therefore, qualitative results from an industrial survey on current practices and preferences, highlighting differences and similarities between development with reusable components, development without reusable components, and development of components for reuse were presented. The results of the study are that the reuse of components does not make permanent design decisions, the verification of components was not being done to a sufficient extent and known good practices for component selection and evaluation were implemented in some organizations but not all. As a conclusion it has been stated that the state of practise of component reuse in industry was that the components were built for reuse and those components were in fact being reused.

6.2 Selection of components

In this section we present the summary of all the studies that focused their research on the area selection of components of CBSE or CBSD. The research mostly focuses on the specification, selection and evaluation of the components before their implementation.

- Metrics for measuring similarity between component interfaces were defined in [21]. As stated in the study the important contributions of the study are:
    - The introduction of software component interface refactorings, i.e., transformations working on IDL-style (i.e., signature-list based) component interfaces. These refactorings are used to define a similarity metric for signature-list based component interfaces.
    - The definition of efficiently computable metrics measuring the similarity of software component protocols (i.e., valid call sequences to component services).
- [22] is a study about the challenges that are involved in specifying the non-functional attributes for components e.g. in terms such as fast, slow, very fast, etc. The study describes the usage of infinite value logic called *Fuzzy logic* in formally specifying such linguistic variables or hedges. In the study, data was collected from components which was fuzzified and represented as membership functions.
- A study has been done on why project decision makers use COTS components instead of OSS components or vice versa [23]. The study was conducted in the form of a survey and data was gathered from international companies in the countries Norway, Germany and Italy. The results were stated as that both COTS and OSS components were used by small, medium and large software companies. It was also stated that the users of COTS components believed that the components should be of good quality, have technical support and follow market trend. On the other hand, OSS components users were stated to be more concerned about the ownership and openness of the source code. Other results stated were that projects using COTS components had more difficulties in estimating selection effort, following customer requirement changes, and controlling the component's negative effect on system security. On the other hand, OSS user had more difficulties in getting the support reputation of OSS component providers.
- Although a multiplicity of COTS selection method have been proposed in literature, most developer still select COTS products using ad hoc methods. One of the main reason being, COTS selection method do not provide all or most of the required support and guidance required for carrying out the COTS selection process. Therefore the study presented in [24] was aimed to find out differences if any, between 3 selection methods and to determine the ability of each of the methods to provide adequate COTS selection support and guidance.
- A study has been conducted to investigate how research can be useful in the process of selection of components [25]. The study is an interview of developers from 16 Norwegian companies which integrate OSS components into their systems. The study reports two results which were stated to be key findings which are:

- Project specific constraints are much more decisive in the selection of OSS components than the general evaluation criteria suggested by existing evaluation schema.
- Software developers employ the principle of 'first fit' as the principle of evaluation, whereas existing research on evaluation and selection methods employs 'best fit'. Rather than identifying a set of components to evaluate, software developers evaluate individual OSS components sequentially. Knowledge gained in rejecting one component is fed back as new evaluation criteria in the evaluation of the next.

- According to [26] there is limited knowledge about the industrial OTS components selection practices, therefore the study investigated the actual industrial practice of component selection in order to provide an initial empirical basis that allows the reconciliation of research and industrial endeavours. The results of the study were stated as that the component repositories were hardly used in the industry whereas the literature claims the repositories are important. Other results were suggestions for researchers to focus their research on the selection practices based on reality instead of assumptions and for software intensive organizations to consider the results of this study which would help them increase their awareness in implication of factors such as experience and knowledge in their selection practices.

6.3 Quality of Components

In this section we present the summary of all the studies that focused their research on the area quality of components of CBSE or CBSD. The research didn't focus only on one or few specific aspects of quality of components instead the focus was on discrete aspects.

- [27] is a study of consumers' preferences and purchasing behaviour of software components regarding to the quality attributes of the components. The study involved a simulation of an artificial marketplace for selling and buying components online. The sellers (producers) and buyers (consumers) were students as the study was conducted as an experiment where the producers had a technical background and the consumers were non-technical. An interesting result, as was stated, was that the popularity of the producer was one of the important preferences of the consumers. Other results were that the consumers preferred components with highly represented functions and sophistication and also pricing and discounts were part of their preferences.

- A study was conducted on software quality evaluation in Korea for its Ministry of National Defence [28]. According to the study, most previous works on software quality evaluation were focused on COTS-based software or deliverable software products with quality model and metrics. However, this study had presented a

quantitative quality evaluation approach with respect to the Component Based Development (CBD) methodology. The approach was such that weights were assigned to quality characteristics based on a questionnaire survey and were processed through a technique called Analytic Hierarchical Process. The result was stated as that the quality evaluation approach was viable and found that the approach is practically possible for use in real projects.

- A study was conducted to predict how failure-prone a component will be in order to facilitate the design decisions [29]. The prediction was based on the components' past failure history and usage relationships. The study was conducted on 52 ECLIPSE plug-ins and the results of the study were stated to support a hypothesis that one can predict future post-release failures by using imported components of a file or package.

- A metric called Component Complexity Metric is proposed which may be used to limit the complexity of the component [12]. According to the study not much work has been done in evaluating quality metrics for components and CBS. An assumption made in the study was that the complexity of a system can be reduced if the components used in the system are not complex. The study was conducted on JavaBeans components and proposed a metric to measure the complexity of those components which was evaluated theoretically by standard Weyuker's properties.

- All the quality attributes may not be of prime importance for a component application, thus a new quality model was proposed in [13] with new characteristics which may be very relevant to the context of components. Similar to the study in [28], this study too had Analytical Hierarchical Process for assigning weights to the quality characteristics and evaluated the quality of a component through an experiment considering a real life example. The result of the study was a quality model with additional quality attributes which can be used for comparison and selection of the best suitable components for the system.

6.4 Reusability of components

In this section we present the summary of all the studies that were conducted in regard to the reusability of components. Most of the studies focused on the quality characteristics of the components with reusability as the backdrop of their works.

- The objective of the study [30] was to evaluate published software metrics that would measure the "time, money and quality" benefits of reuse of components. The study was conducted in two ways – analytically and empirically. For analytical evaluation, some desirable properties of reuse benefit measures were proposed and evaluated the metrics in terms of their compliance with the properties. The result of this analytical evaluation was stated to be that none of the properties satisfied all the properties proposed. For empirical evaluation of the metrics, a toolset was

constructed to gather data on all published reuse metrics from C++ code and finally verified statistically the correlation between the metrics and the quality factors of productivity and defect density. The result of this empirical evaluation was stated to be that different reuse metrics can be used as predictors of different quality attributes. An example was stated for this as, reuse ratio and size/frequency reuse metric each appeared to be well correlated with productivity and error density, but this size/frequency metric did not show any significant result with regard to fault density.

- Component Reuse Metrics, CRM, was a new effort estimation method, which considers software development as a series of tasks of assembling software components. CRM adds assessments of project and human effects of the development project to the component-based effort estimates. The approach of this study was empirical and the main result of this paper [31] was an evaluation of the CRM method by a survey and by case studies. The result was stated to be that the case studies has confirmed that component-based development creates an acceptable component structure for CRM calculations. It was stated that the assessment of project and human effects proved to be difficult at least without experience and historical data.

- An active reuse repository system called *CodeBroker* was used to show that active repository systems promote reuse by motivating and enabling software developers to reuse components whose existence is not anticipated, and reducing the cost of reuse through the automation of the component location process [32].
- Hypotheses about impact of reuse on defect density and stability and impact of component size on defects and defect density in the context of reuse were assessed in [33]. The results were stated as follows:
    - *Reuse and defect-density* – The result was that the reused components have lower defect density than non-reused ones and the difference was less for modified code. An observation mentioned was that the reused components had more severity A defects than expected from the total distribution, but fewer post-delivery defects.
    - *Number of defects and component size* – The result was that no relation was observed between number of defects and component size.
    - *Defect density and component size* – The result was stated as that the plots and regression analysis did not show any relation between defect density and component size.
    - *Reuse and stability* – The result was stated to be that when the components were reused across several products, they got more fragile.
- [34] discusses the modularity offered by Aspect-Oriented Programming and its association with obliviousness and the trade-offs between modularity and

obliviousness. The study presents a refactoring of exception handling concern for three real-life Java applications to use explicit joint points (EJPs) instead of oblivious aspects. The empirical differences between this version and an equivalent oblivious version were analyzed. By using EJPs the obliviousness from an aspect-oriented implementation of a cross-cutting concern was removed, and then the resulting effects on software quality and modularity were studied. The results were stated to be as follows:

- The use of an explicit interface to model cross-cutting concerns facilitates the creation of reusable aspect libraries.
- The parameterization of aspects made possible by these explicit interfaces can increase code reuse and reduce point-cut complexity.
- Explicit cross-cutting interfaces must be carefully designed to be as minimal as possible or overall application modularity may actually decrease.
- The greatest application modularity is achieved when a combination of explicit join points and oblivious aspects are applied. When pointcuts can be written in a stable fashion they are favored over explicitness in the base code, but in the remaining cases using EJPs result in better code quality.

6.5 CBSD Process

In this section we present the summary of all those studies that were conducted in regard to the CBSD or CBSE process.

- Human, social and organisational issues affecting the introduction of Component-Based Development (CBD) in organizations were presented in [35]. The result of the study was stated to be as follows:
  - Encouraging customer or user participation
  - Encouraging interaction of users and developers
  - Educating all stakeholders about the whole CBD process
  - Using incremental approach

  An important lesson learnt, as was stated in the study, was that integrating organizational issues in the CBD process was a difficult problem requiring different strategies depending on the organizational context. However, social–technical approaches could help to incorporate and resolve organizational obstacles by encouraging user participation in CBD.

- The work described in this paper is an investigation of the COTS-based software development within a particular NASA environment, with an emphasis on the processes used [36]. Fifteen projects using a COTS-based approach were studied and their actual process was documented. This process was evaluated to identify essential differences in comparison to traditional software development. A new

process and set of guidelines for COTS based development were developed and briefly presented.

- Although previous studies have proposed specific COTS-based development processes, there are few empirical studies that investigate how to use and customize COTS-based development processes for different project contexts. This paper describes an exploratory study conducted in 16 Norwegian IT companies through structured interviews, of state-of-the-practice of COTS-based development processes [37]. The results were stated as that the COTS-specific activities can be successfully incorporated in most traditional development processes (such as waterfall or prototyping), given proper guidelines to reduce risks and provide specific assistance. 4 COTS-specific activities were identified – The build vs. buy decision, COTS component selection, learning and understanding COTS components, and COTS component integration – and one new role, that of a knowledge keeper. 2 component selection processes were also discovered – familiarity based and combining Internet search with hands-on trials

- According to [38] CBD will improve globally distributed software development practices by allowing each site to take ownership of particular components, resulting in reduced inter-site communication and coordination activities. It was stated that such an approach may indeed overcome breakdowns in inter-site coordination efforts; however, it may also lessen opportunities to share knowledge between sites and may hamper opportunities to reuse existing components. A case study approach, exploratory in nature, was adopted to explore knowledge aspects in global component-based software development. The result stated was that the true potential of CBD, which mainly relates to reuse of components, can be achieved through sharing of expertise of the teams irrespective of their geographical location. Finally some guidelines to managers and engineers were presented.

6.6 Performance of components

In this section we present the summary of all studies that were conducted with respect to the performance of components of a CBS.

- A scalable security mechanism named *Component Adaptive Scalable Secure Infrastructure Architecture* (CASSIA), for component based systems is proposed [11]. The result of the study was stated as that 80% of the components in real world CBS are used in a protected security perimeter. A case study was performed to confirm the scalability of CASSIA and a protocol named Secure COmponent Protocol (SCOP) was proposed which uses CASSIA inside a Large-Scale component infrastructure.

- The actual effort required for developing a performance prediction model is addressed by proposing a component-based prediction model named Palladio in [39]. An experiment was conducted comprising 19 computer science students to apply the Software Performance Engineering method and the Palladio method to predict the performance of two example systems. The result was stated as that the effort for applying Palladio on the whole task was in average 1.25 times the effort for applying SPE. It was suggested to put more focus on research about creating reusable models as their creation can quickly yield results.

- Relation between autonomy and qualities of the system was studied by proposing an approach for quantifying autonomy in [40]. The approach was such that a mathematical model was built and a Traffic Control Simulation System was built based on the mathematical model and comprising autonomous components. An experiment was conducted on this system to explore the relationship between the autonomy of these components and their quality characteristics. The results were stated as that –

    - Certain autonomy is necessary for the system
    - The relation between autonomy and quality property is not monotonic i.e. increase in autonomy of the component may decrease its quality
    - Different environments are associated with different level of autonomies, which eventually impact the quality property. With less complex environment autonomy has low correlation with quality which means the autonomy could be raised for user's convenience.

6.7 Component Testing

In this section we present the summary of all the studies that were conducted with respect to component testing.

- [41] addresses the issue of usability testing in a component based software engineering environment, specifically measuring the usability of different versions of a component in a more powerful manner than other, more holistic, usability methods. Three component-specific usability measures were presented: an objective performance measure, a perceived ease-of-use measure, and a satisfaction measure. The result in the study confirms the possibility of testing the usability of individual components, which can be applied in a CBSE environment.

- [42] presents an approach to support component testing aiming to reduce the lack of information between component producers and consumers. Two workflows were presented describing necessary activities to be conducted by producers to prepare a component to be tested by third party; and the activities performed by component consumers to elaborate and execute test cases to support the decision of integrating candidate components to a system under development. A formal experimental study

was performed in order to evaluate the viability of applying the proposed component testing approach, as well as its tool support.

- [43] describes a first empirical study comparing two defect detection techniques – code inspections and functional testing in the context of product line development of reusable components. The primary goal of the study was to initially investigate the defect finding potential of the techniques on reusable software components with common and variant features. The major findings of the study are that the two techniques identified different types of defects on variants of a reusable component. The study not only investigated the efficiency and effectiveness of the two techniques but also found that what types of defects can be detected in common and product-specific parts of a reusable component. The result of an experiment conducted in the study was stated to support the hypothesis that different techniques identify different types of defects and have different efficiency and effectiveness.

6.8 Storage of components

In this section we present the summary of all the studies that were conducted with respect to storage of components.

- The study [44] proposed a new scheme for classifying and describing business components and the prototype version of the knowledge based repository for storage and retrieval of components. The study consists of an experiment that has been conducted with an average age group of participants. The results of the experiment confirm that a formal mechanism for classifying, coding & storing components in knowledge based repository enhances analysts (assemblers) ability to find the required business components.

- In study [45] different component indexing & retrieval methods were tested and found that full text indexing and retrieval of software components is better than controlled vocabulary indexing and retrieval. The study attempted to bring two kinds of methods to a level playing field by addressing the cost issues and using a realistic experimental setting which includes realistic evaluation measures. In this study there were total of three experiments trying out different ideas and comparing approaches. The results showed that the aspects of the pre-processing involved in controlled vocabulary methods that they automated were of poor quality that were not used and the fully automatic free text search performed better than the fully manual controlled vocabulary based indexing and retrieval of components.

6.9 Integration of components

In this section we present the summary of all the studies that were conducted with respect to integration of components.

- In this study [46] the process of glue code generation and the process of integration of components were discussed through software simulation that provides a method for checking the understanding of the real world process. This study also reports simulation results showing how different concurrency profiles affect staffing levels and how various starting points of glue code development have an effect on system integration processes. It also suggests another effective strategy from a schedule reduction point of view and illustrates both utility and limitations of modelling in general.
- The study [47] reports results of an industrial survey conducted among system integrators to understand role of component documentation in the CBS integration phase. The survey investigates whether the presence of component documentation helps a system integrator and its correlations with typical integration success factors. The results indicate that available component documentation class help in integrating selected components. The participants of the survey found that available component documentation was useful in early component evaluation and discovering new features. However, on an average they have found that component documentation does not provide enough information to overcome the two most common CBS integration challenges – incorrect integration effort estimation and integration testing.

6.10 Implementation and Maintenance of components

In this section we present the summary of all the studies that were conducted with respect to Implementation and Maintenance of components.

- In this study [48] different level of components reuse and certain aspects of component development like component generality and efficiency, compatibility problems, the demands on development environment and maintenance were discussed. The evolution of requirements for products generates new requirements for components if components are not enough general and mature. This dynamism determines the component life cycle where the component first reaches its stability and later degenerates in an asset that is difficult to use, difficult to adapt and maintain. When reaching this stage, the component becomes an obstacle for efficient reuse and should be replaced. Questions related to use of standard and de-facto standard components are addressed specifically and this study also presents a successful implementation of a component-based system which is widely used for industrial process control. The case study was done in ABB with an advent control

system as an example for component-based system. It was stated that the success of this system on the market has been primarily the result of appropriate functionality and quality. The study found that the organization was successful because of its systematic approach in design planning, development & maintenance.

- This study [49] is a part of the previous study. In this study the issues and challenges that are encountered when developing and using an evolving component-based software system were discussed by doing a case study. The results of the study presents many practical issues that were noticed while designing the components and incorporating them in to the system.

6.11 Design and Implementation of components

In this section we present the summary of all the studies that were conducted with respect to design and implementation of components.

- In this study [50] the best practices in designing and building a web based auction system by using UML and components were presented through an empirical study. The study describes a case study in which a web auction system using UML and components was implemented. The rigorous design and analysis phase and the robust component based implementation enabled them to achieve a minimal defect rate in the final product. The defect rate of the reused code was 0.9 units per 1 KLOC. The scope of implementation and identification of entities that could be coded as reusable components was done with the help of UML. The implementation with its basis in component based programming enabled them to develop a highly maintainable system with a number of reusable components.

- In this study [51] two critical aspects of component based systems in the financial industry were addressed. Component based design of systems and the mediation between the components were the two aspects. This study uses examples in the industry to demonstrate component-based implementation and semantics mediation. To investigate the fundamental mechanism for further understanding of the meaning of semantics mediation benchmarking was executed by using both empirical experiments and theoretical modelling. The observations show that a significant enhancement can be achieved by using the mediation strategy in semantic metadata.

6.12 Component Architecture

In this section we present the summary of all the studies that were conducted with respect to Component Architecture.

- [52] presents a case study from a global company developing a new generation of programmable controllers to replace several existing products. The study also

presents that advantages and liabilities, the use of a components based software architecture entails for the development of an industrial control system are presented. The results from the study showed that the effort required adding support for communication protocols in the controller product has been considerably reduced due to the adoption of new architecture (component –based software architecture).

- According to [53] Component-based software development needs to formalize a process of generation, evaluation and selection of composite COTS- based software system (CCSS), enabling software architects to make early decisions; the Azimut approach and its associated software tool were proposed to tackle this problem. This article presents an experimental study conducted to compare Azimut approach with a systematized Ad-hoc approach, regarding generated solutions quality, cost & effort. It also serves a frame-work for validating approaches, process & tools for generating & evaluating component-based software systems. The results suggest that Azimut generated better quality solutions at lower cost, although not statistically significant in three samples, but also there is strong statistically significant evidence showing that the effort required is higher than for Ad-hoc. Re-sampling methods were applied to reinforce these conclusions and yielded the same results. Results concerning effort are aligned with a post experiment survey answered by the participants where they suggest various ideas for improving the usability of Azimut tool user interface.

6.13 Maintenance of components

[54] is an experiment investigating component collaborations in the OSGI/ Eclipse component model. The main aim of the experiment was to demonstrate the benefits of using a formal contract language for dynamically composed systems. According to the study, these systems are no longer assembled, but instead they evolve while deployed. Components of such systems are replaced while system remains operational making it an important task of the maintenance routine. For this study, the data was obtained by measuring metrics for these contracts and by verifying a large eclipse distribution against these contracts. The outcome of this study supported the claim that it is useful to employ a formal contract language.

6.14 Extensibility of components

According to [10], it's not an easy task to identify the minimal changes required in an evolving or extensible CBS and mitigate the possible side-effects caused by those changes. This study focused on three heuristics and compared their properties on an evolving GNU/Linux distribution. The aim was to find a heuristic to minimize change by following a process called Component Dependency Resolution. The results of the study were stated as follows:

- There is a high probability a set of user-requests is not satisfiable.
- Most user-requests cause little change, however a few requests require significant change to the system.
- Complex heuristics (like PageRank) have little difference from greedy, simple heuristics (like Hamming) and take longer to and a solution.

7. Limitations of this study

7.1 Completeness

We have thoroughly extracted most of the empirical studies that were published between 1995 and August 2011 that were conducted on CBSD or CBSE or CBS. However, we still feel that we might have missed few studies because of the reason that they might have been published in journals or conferences or research articles which may had not gained much attention from the research community. Therefore we considered this as one of our limitations.

7.2 Data synthesis

We gathered all the studies and tabularized the data from these studies such as the year of publication, the authors, the description, etc. for assessing quality and finally answering our research questions. We were simply guided by this table in the analysis process in which we identified the topics that were less researched with respect to the methodologies that were followed in the respective studies. The literature mostly discussed the research topics as a combination of the topics such as the one done in [33]. This study discusses the topic of reusability of components and also the issue of defect density at the same time. We grouped this type of studies under the category or the topic that is given much attention or is the core issue that was being discussed in that respective study. In the example of [33] we grouped it under the category of reusability of components though it discusses another issue in parallel. From this we would like to say that we might have missed some studies which might have come under a new category which means that a new category would have been added to the existing ones and group the studies under the new one.

7.3 Potential bias

Since this study has been conducted by a single researcher, there is always scope for the researcher's bias in the quality assessment of the studies and also in the discussion of the answers for the research questions.

8. Future work

The future work to this study could possibly be done towards extracting more analytical information from all the studies by attempting to reduce the limitations mentioned so far. This could be done in such a way that another two to four researchers could join the work and repeat the procedure mentioned in the method section of this study. This would finally lead to more unbiased results and produces more analytical information thereby adding additional value to the existing data presented in the discussion section.

9. Conclusion

We have presented a systematic literature review of all the empirical studies done with respect to CBSD process. We have reviewed all the studies that were published between January 1995 and August 2011. We found that the empirical studies have covered most of the phases of CBSD in their research.

We extracted 47 studies which is the amount of empirical research that has been done on CBSD through the period that is mentioned above. *Case studies* and *Experiments* were the most preferred research methods which constitute 42.5% and 40.4% of the total number of studies respectively. About the research topics that were discussed, *Implementation of Components*, *Selection of Components* and *Quality of Components* were highly discussed with the frequency of 7, 6 and 5 studies respectively.

We found that, among case studies, industrial case studies were the most prevalent and a preferred research methodology. There were 10 industrial case studies out of the total 19 case studies. Regarding the geographical distribution of the industrial research, much of it has been done in Europe with 9 studies and 1 study in China. We highlight that the industrial research as a factor that could be improved in terms of its global persistence keeping in mind the benefits of a socio-economic and business environment.

Regarding experiments, we presented the observation of [3] about the student subjects in experiments that which could be improved by choosing professionals or industry practitioners as the subjects so that the results can be made more generalizable.

Apart from the above observations we had identified certain topics or subjects that were not focused in the empirical studies. No empirical studies were available:

- justifying the claim of CBSE or CBSD process that it reduces development costs and also the product's time to market
- presenting new knowledge about component models either on existing ones like COM, CORBA or proposing new models
- showing the current-state-of-practice on the usage of CASE tools or development environments that facilitate the CBSE or CBSD practice

Finally, we presented a comprehensive summary of all the collected studies according to the research topics they discussed.

10. References

1. Jerry Gao, H.-S.J.T.Y.W., *Testing and Quality Assurance for Component-Based Software*. 2003, Norwood, MA, USA: Artech House Publisher.

2. Barbara Kitchenham, S.C., *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. 2007, Keele University: Keele, UK.

3. Höfer, A. and W. Tichy, *Status of Empirical Research in Software Engineering Empirical Software Engineering Issues. Critical Assessment and Future Directions*, V. Basili, et al., Editors. 2007, Springer Berlin / Heidelberg. p. 10-19.

4. Kitchenham, B., et al., *Systematic literature reviews in software engineering – A systematic literature review.* Information and Software Technology, 2009. **51**(1): p. 7-15.

5. Beecham, S., et al., *Motivation in Software Engineering: A systematic literature review.* Information and Software Technology, 2008. **50**(9-10): p. 860-878.

6. Karam, F.T.O., *Essentials of Software Engineering*. 2009, Sudbury, MA, USA: Jones and Bartlet Publishers.

7. Councill, G.T.H.W.T., *Component-Based Software Engineering: Putting the pieces together*. 2001, Boston, MA, USA: Addison-Wesley.

8. Clemens Szyperski, D.G.S.M., *Component Software: Beyond Object-Oriented Programming*. 2nd ed. 2002, London: Addison-Wesley and ACM Press.

9. Crnkovic, I., M. Chaudron, and S. Larsson. *Component-Based Development Process and Component Lifecycle*. in *Software Engineering Advances, International Conference on*. 2006.

10. Jenson, G., et al., *An empirical study into component system evolution*, in *Proceedings of the 14th international ACM Sigsoft symposium on Component based software engineering*. 2011, ACM: Boulder, Colorado, USA. p. 189-192.

11. Grechanik, M., D.E. Perry, and D. Batory. *A security mechanism for component-based systems*. in *Commercial-off-the-Shelf (COTS)-Based Software Systems, 2006. Fifth International Conference on*. 2006.

12. Sharma, A., R. Kumar, and P.S. Grover, *Empirical evaluation and critical review of complexity metrics for software components*, in *Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*. 2007, World Scientific and Engineering Academy and Society (WSEAS): Corfu Island, Greece. p. 24-29.

13. Sharma, A., R. Kumar, and P.S. Grover, *Estimation of quality for software components: an empirical approach*. SIGSOFT Softw. Eng. Notes, 2008. **33**(6): p. 1-10.

14. Ishikawa.H, N.T. *A Case study on a Component Based System and its Configuration*. in *In proceedings of SCOPES 2003*. 2003.

15. Kvale, A.A., J. Li, and R. Conradi, *A case study on building COTS-based system using aspect-oriented programming*, in *Proceedings of the 2005 ACM symposium on Applied computing*. 2005, ACM: Santa Fe, New Mexico. p. 1491-1498.

16. Stevens, G., G. Quaisser, and M. Klann, *Breaking It Up: An Industrial Case Study of Component-Based Tailorable Software Design End User Development*, H. Lieberman, F. Paternò, and V. Wulf, Editors. 2006, Springer Netherlands. p. 269-294.

17. Ajila, S.A. and D. Wu, *Empirical study of the effects of open source adoption on software development economics*. Journal of Systems and Software, 2007. **80**(9): p. 1517-1529.

18. Slyngstad, O.P.N., et al. *The Impact of Test Driven Development on the Evolution of a Reusable Framework of Components – An Industrial Case Study*. in *Software Engineering Advances, 2008. ICSEA '08. The Third International Conference on*. 2008.

19. Chen, W., et al., *A Survey of Software Development with Open Source Components in Chinese Software Industry Software Process Dynamics and Agility*, Q. Wang, D. Pfahl, and D. Raffo, Editors. 2007, Springer Berlin / Heidelberg. p. 208-220.

20. Land, R., et al., *Reuse with Software Components - A Survey of Industrial State of Practice Formal Foundations of Reuse and Domain Engineering*, S. Edwards and G. Kulczycki, Editors. 2009, Springer Berlin / Heidelberg. p. 150-159.

21. Kratz, B., R. Reussner, and W.-J.v.d. Heuvel, *Empirical Research Similarity Metrics For Software Component Interfaces*. J. Integr. Des. Process Sci., 2004. **8**(4): p. 1-17.

22. Cooper, K., et al., *An Empirical Study on the Specification and Selection of Components Using Fuzzy Logic Component-Based Software Engineering*, G. Heineman, et al., Editors. 2005, Springer Berlin / Heidelberg. p. 18-20.

23. Li, J., et al., *An empirical study on decision making in off-the-shelf component-based development*, in *Proceedings of the 28th international conference on Software engineering*. 2006, ACM: Shanghai, China. p. 897-900.

24. Far, T.W.B.H., *An Empirical Study to Compare Three Methods for Selecting Cots Software Components* International Journal of computing & ICT Research, 2008. **2 NO 1**: p. 34-46.

25. Hauge, O., et al. *An empirical study on selection of Open Source Software - Preliminary results*. in *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on*. 2009.

26. Ayala, C., et al., *Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners*. Journal of Systems and Software, 2011. **84**(4): p. 620-637.

27. Hong, S.-J. and F.J. Lerch, *A laboratory study of consumers' preferences and purchasing behavior with regards to software components.* SIGMIS Database, 2002. **33**(3): p. 23-37.

28. Lee, K. and S. Lee, *A Quantitative Evaluation Model Using the ISO/IEC 9126 Quality Model in the Component Based Development Process Computational Science and Its Applications - ICCSA 2006*, M. Gavrilova, et al., Editors. 2006, Springer Berlin / Heidelberg. p. 917-926.

29. Schr\, A., et al., *Predicting component failures at design time*, in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. 2006, ACM: Rio de Janeiro, Brazil. p. 18-27.

30. Devanbu, P., et al., *Analytical and empirical evaluation of software reuse metrics*, in *Proceedings of the 18th international conference on Software engineering*. 1996, IEEE Computer Society: Berlin, Germany. p. 189-199.

31. Virtanen, P. *Empirical Study Evaluating Component Reuse Metrics*. in *Proceedings of the ESCOM 2001*. 2001.

32. Ye, Y., *An Empirical User Study of an Active Reuse Repository System*, in *Proceedings of the 7th International Conference on Software Reuse: Methods, Techniques, and Tools*. 2002, Springer-Verlag. p. 281-292.

33. Mohagheghi, P., et al., *An Empirical Study of Software Reuse vs. Defect-Density and Stability*, in *Proceedings of the 26th International Conference on Software Engineering*. 2004, IEEE Computer Society. p. 282-292.

34. Hoffman, K. and P. Eugster, *Towards reusable components with aspects: an empirical study on modularity and obliviousness*, in *Proceedings of the 30th international conference on Software engineering*. 2008, ACM: Leipzig, Germany. p. 91-100.

35. Kunda, D. and L. Brooks, *Assessing organisational obstacles to component-based development: a case study approach.* Information and Software Technology, 2000. **42**(10): p. 715-725.

36. Morisio, M., et al., *COTS-based software development: Processes and open issues.* Journal of Systems and Software, 2002. **61**(3): p. 189-199.

37. Li, J., et al., *An empirical study of variations in COTS-based software development processes in the Norwegian IT industry.* Empirical Software Engineering, 2006. **11**(3): p. 433-461.

38. Julia Kortlarsky, I.O., Jos van Hillegersberg, Kuldeep Kumar, *Globally distributed component-based software developement:an exploratory study of knowledge management and work division.* Information Technology, 2007. **22**(2): p. 161-173.

39. Martens, A., et al., *An Empirical Investigation of the Effort of Creating Reusable, Component-Based Models for Performance Prediction Component-Based Software Engineering*, M. Chaudron, C. Szyperski, and R. Reussner, Editors. 2008, Springer Berlin / Heidelberg. p. 16-31.

40. Tingxun, S., et al. *An Empirical Study on the Impacts of Autonomy of Components on Qualities of Software Systems*. in *Engineering of Autonomic and Autonomous Systems (EASe), 2011 8th IEEE International Conference and Workshops on*. 2011.

41. Brinkman, W.-P., R. Haakma, and D. Bouwhuis, *Empirical Usability Testing in a Component-Based Environment: Improving Test Efficiency with Component-Specific Usability Measures Engineering Human Computer Interaction and Interactive Systems*, R. Bastide, P. Palanque, and J. Roth, Editors. 2005, Springer Berlin / Heidelberg. p. 880-880.

42. Silva, F.R.C., E.S. Almeida, and S.R.L. Meira, *An approach for component testing and its empirical validation*, in *Proceedings of the 2009 ACM symposium on Applied Computing*. 2009, ACM: Honolulu, Hawaii. p. 574-581.

43. Denger, C. and R. Kolb, *Testing and inspecting reusable product line components: first empirical results*, in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. 2006, ACM: Rio de Janeiro, Brazil. p. 184-193.

44. Vitharana, P., F.M. Zahedi, and H. Jain, *Knowledge-based repository scheme for storing and retrieving business components: a theoretical design and an empirical analysis.* Software Engineering, IEEE Transactions on, 2003. **29**(7): p. 649-664.

45. Mili, H., et al., *An experiment in software component retrieval.* Information and Software Technology, 2003. **45**(10): p. 633-649.

46. Baik, J., N. Eickelmann, and C. Abts. *Empirical software simulation for COTS glue code development and integration*. in *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*. 2001.

47. Mahmood, S. and A. Khan, *An industrial study on the importance of software component documentation: A system integrator's perspective.* Information Processing Letters, 2011. **111**(12): p. 583-590.

48. Crnkovic, I. and M. Larsson, *A case study: demands on component-based development*, in *Proceedings of the 22nd international conference on Software engineering*. 2000, ACM: Limerick, Ireland. p. 23-31.

49. Crnkovic, I. and M. Larsson, *Challenges of component-based development.* Journal of Systems and Software, 2002. **61**(3): p. 201-212.

50. Sheldon, F.T., et al. *Case study: implementing a web based auction system using UML and component-based programming*. in *Computer Software and Applications Conference, 2002. COMPSAC 2002. Proceedings. 26th Annual International*. 2002.

51. Wu, R., *An industry case study of micro component design and semantic mediation.* Int. J. Metadata Semant. Ontologies, 2007. **2**(4): p. 223-234.

52. Lüders, F., I. Crnkovic, and P. Runeson, *Adopting a Component-Based Software Architecture for an Industrial Control System – A Case Study Component-Based Software Development for Embedded Systems*, C. Atkinson, et al., Editors. 2005, Springer Berlin / Heidelberg. p. 232-248.

53. Victor Sagredo, C.B.G.V. (2010) *Empirical Validation of Component Based Software Systems Generation and Evaluation Approaches*. Clei. electronic. jounal **13**, 13.

54. Dietrich, J. and L. Stewart, *Component Contracts in Eclipse - A Case Study Component-Based Software Engineering*, L. Grunske, R. Reussner, and F. Plasil, Editors. 2010, Springer Berlin / Heidelberg. p. 150-165.

55. Runeson, P. and M. Höst, *Guidelines for conducting and reporting case study research in software engineering*. Empirical Software Engineering, 2009. **14**(2): p. 131-164.

56. Yakimovich, D., Travassos, G. H., and Basili, V., *A classification of software components incompatibilities for COTS integration*. 2000.

57. Sun, C., *Empirical reasoning about quality of service of component-based distributed systems*, in *Proceedings of the 42nd annual Southeast regional conference*. 2004, ACM: Huntsville, Alabama. p. 341-346.