Measuring Cost Avoidance Through Software Reuse

A model to measure costs avoided through software reuse and guidelines to increase profits through software reuse

Mohsin Irshad

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden
This thesis is submitted to the School of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author:
Mohsin Irshad 840930-1531
E-mail: node.node@gmail.com

Industrial advisor:
Martin Wallin
Ericsson AB, Karlskrona, Sweden

University advisor:
Prof. Richard Torkar
School of Computing

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Internet : www.bth.se/com
Abstract

**Context.** Software Reuse is considered as silver-bullet for software development. However, measuring benefits of software reuse is difficult and cumbersome task because of varying number of factors involved in it. Different reuse cost models already exist in literature which measure various different attributes of software reuse. Mainly these models are used for calculating return over investment or cost-benefit analysis.

**Objectives.** We have investigated that very few cost economic models have been proposed for measuring costs avoidance, degree of empirical validation, assumptions, types of artifacts they can measure and whether they provide guidelines on collection of metrics for measuring reuse benefits.

**Methods.** In this research, a systematic review was conducted. Based on the results of systematic review, a model was proposed which can measure cost avoided by reuse of every kind of artifact. In a systematic all major article sources were used. Studies were selected after reading titles and abstracts. Three cost avoidance models were found and an analysis of these models was performed. Based on the analysis, a new model was proposed to fill the gap left by these studies.

**Results.** New model measures every kind of reuse artifact and provides guidelines on how and what to measure in order to calculate reuse benefits. This model was then validated in the industry and technology was transferred to the industry for future usage. Guidelines for improved savings were developed.

**Conclusions.** We conclude that many models are related to each other and use similar techniques to measure the cost avoidance however they can not measure all kinds of reuse artifacts. New model performed well in industry. However, we found that the new model should accommodate maintenance costs since these are major savings by software reuse. Moreover, we conclude that there is a need for further validation of guidelines and model in industry.

**Keywords:** Cost Avoidance, Software Reuse, Maintenance Costs, Dynamic Validation, Guidelines, Savings, Benefits
## Contents

Abstract .................................................. i

1 Introduction ........................................... 1

2 Related Work (Systematic Literature Review) ....... 3
   2.1 Generation of search strategy ...................... 4
      2.1.1 Search Terms Construction .................. 4
      2.1.2 Electronic Database .......................... 4
   2.2 Study selection criteria .......................... 6
      2.2.1 Exclusion Criteria ........................... 6
      2.2.2 Inclusion Criteria ............................ 7
   2.3 Quality Assessment Criteria ....................... 7
   2.4 Data Extraction .................................. 8
   2.5 Conducting the review ............................ 8
   2.6 Analysis and Discussion ........................ 12
      2.6.1 Available Models for Cost Avoidance: ....... 12
      2.6.2 Metrics required .............................. 14
      2.6.3 Limitations of Models ......................... 15
      2.6.4 Validation in Industry ......................... 15
      2.6.5 Quality Assessment ............................ 16
   2.7 Conclusions and Discussion ....................... 16
   2.8 Limitations ....................................... 17
   2.9 Summary of Literature Review ..................... 18

3 Background Work ....................................... 19
   3.1 A Survey on an organizations reuse process: Results and Analysis 19
      3.1.1 Research objectives and methodology ........ 19
      3.1.2 Questionnaire ................................ 20
      3.1.3 Data Collection ............................... 21
      3.1.4 Results Obtained .............................. 21
      3.1.5 Patterns of Reuse ............................. 22
      3.1.6 Cost Factor .................................. 25
      3.1.7 Interpretation and analysis of results ....... 26
      3.1.8 Recommendations .............................. 27
3.1.9 Future work ..................................................... 28
3.1.10 Validity .......................................................... 29
3.1.11 Conclusion ....................................................... 29
3.2 Literature Survey on Reuse Practices of 5 Large organizations .. 30
   3.2.1 Industrial Review .............................................. 30
   3.2.2 Analysis and Discussion ................................... 37
   3.2.3 Recommendations ............................................ 39
   3.2.4 Validity ........................................................ 39
   3.2.5 Conclusion ..................................................... 40
3.3 Reuse of a black-box component An Industrial Experience ....... 40
   3.3.1 Factors and Discussion ...................................... 41
   3.3.2 Recommendations ............................................. 45
   3.3.3 Future work ................................................... 45
   3.3.4 Conclusion ..................................................... 46

4 Research Methodology ................................................. 48
   4.1 Research Design ................................................ 48
   4.2 Conducting the Study .......................................... 49
      4.2.1 Phase 1: Preliminary investigation ...................... 50
      4.2.2 Phase 2: Model Creation .................................. 51
      4.2.3 Phase 3: Testing in industrial settings .................. 53
      4.2.4 Phase 4: Release of complete solution .................. 55

5 Model for calculating Cost Avoidance ................................ 56
   5.1 Software reuse metric plan ................................... 56
   5.2 The Model ....................................................... 57
      5.2.1 Third party Reuse products ................................ 57
      5.2.2 Hourly Rate ................................................ 57
      5.2.3 Guidance in data collection ............................... 58
   5.3 Discussion ....................................................... 58
      5.3.1 Coverage ................................................... 58
      5.3.2 Number of reuse instances ................................ 58
      5.3.3 Cost Time relationship ..................................... 59
      5.3.4 Productivity ................................................ 59
      5.3.5 Validity of Model on Different Reuse Artifacts ........... 59
      5.3.6 Generic Model .............................................. 59
      5.3.7 Risks Associated .......................................... 60
   5.4 Dynamic Validation .............................................. 60
      5.4.1 Initiation ................................................... 61
      5.4.2 Data Analysis .............................................. 62
      5.4.3 Results ..................................................... 62
      5.4.4 Evaluation of Model ....................................... 63
6 Guidelines for introducing reuse culture

6.1 Initial Strategy

6.1.1 Small scale domain analysis

6.1.2 Introduction of component library

6.1.3 Contact person for reusable artifact

6.2 Short-term Strategy (1 to 3 years time)

6.2.1 Encourage Object Oriented based design

6.2.2 Optimize component library

6.2.3 Store medium size components

6.2.4 Encourage black-box reuse

6.2.5 Design applications keeping in mind their reuse

6.2.6 Documentation, groups, forums on reuse

6.2.7 Incentive on Reuse

6.3 Long-term Strategy (3 to 5+ years time)

6.3.1 Development language and framework

6.3.2 Refactoring Contents of Component Library

6.3.3 Product Line Engineering

7 Conclusion

A Appendix A

B Appendix B

C Appendix C

D Appendix D

E Definitions

References
Software organizations are always looking for methods in order to reduce costs, improve quality and decrease development time. Software reuse has been focus of most organizations as it can help in decreasing cost, achieving quality product and reducing time to market a product. However, introducing reuse culture in an organization is difficult because software reuse is concerned with various aspects of software development. Technical, management and business aspects of software reuse are still under investigation and myths related to software reuse are described in literature [67]. There are no agreed upon guidelines which can help organizations in introduction of reuse culture, measuring benefits through reuse, using methods which can measure amount of software reuse present inside the organization and in solving other problems related to software reuse. In software engineering software reuse has been claimed as an important source of saving costs [7]. Activities in software reuse vary from management aspects to complex development aspects, increasing its complexity and value to the software development process. Software Reuse can be an important source of avoiding costs and to increase savings for any software development organization.

Measuring costs saved through software reuse is challenge because of its dependency on management, technical and organizational factors. Variety of software reuses cost models already exist in literature [39] [53]. These models were categorized by Poulin [53] into three different categorize. These categorize were based on the nature of these models and type of problems they address. According to Poulin [53] reuse economic models can be divided into three main types: Return on Investment models (ROI models), Cost Benefit Analysis models and Cost Avoidance models. These models look similar but they provide solution to different problems. Return on Investment models are applicable on organizations which have already invested in reuse processes and now trying to measure the benefits they have gained. Cost-Benefit models are used for making decisions on reuse investments. These types of models are useful for management as they help in making optimal decisions before any artifact is reused. Cost avoidance models are least discussed in literature. They have limited scope as they can only be applied after the artifact has been reused and are simple enough to apply. An important factor in cost avoidance is that it does not impose any restriction on
Chapter 1. Introduction

reuse practices. Cost avoidance does not require any investment up front in order to yield benefits; organizations can measure benefits through reuse without any major initial investments. Benefits are measured regardless of investment done in software reuse.

In this study, aim was to find a method which can help organizations in measuring cost avoidance and provide them guidelines for increasing profitability without major changes in organization structure. As evident from literature review, present in Chapter 2, measuring benefits through all kinds of software reuse artifacts and guidelines on how and what to measure are missing. Literature and industrial experiences usually focus on code reuse only and calculate costs avoided through code reuse. In this study we wanted to develop method to measure cost avoided through every kind of reuse artifact. Apply our developed method in any large organization and analyze the results.

Therefore this study is focused on cost avoidance models which can measure software reuse and guidelines which can improve profitability through software reuse. A systematic literature review was conducted on cost avoidance models in software reuse. Results from this review showed three different models which support measuring of cost avoidance through software reuse. Observations made by author inside a very large telecommunication organization helped in producing guidelines for increasing profits. These guidelines are in the form of step-wise process. Further description of systematic review is present in Chapter 2.

In Chapter 2 of the study already existing models are described. Their flaws and limitations are also discussed. Chapters 2 and 3 discuss need of new model. In Chapter 5 a new model is proposed which tries to address the issues found in previous models. Same chapter discusses validation of new model in industry. Chapter 6 discusses guidelines for increasing profitability through software reuse. Conclusions and suggestions are summed up later.
Systematic literature review helps in identification and analysis of studies relevant to this topic. Kitchenham described steps for performing the systematic literature review [35]. Benefits of a systematic literature review were listed by Kitchenham [35]. In order to carry out this study, steps defined by Kitchenham were followed. This section describes all the necessary steps related to our study, as outlined by Kitchenham [35]. Kitchenham’s method was selected because it was widely used by research community. Each step is well defined and many systematic reviews are based on this method.

In order to analyze the pros and cons of reuse cost avoidance model, following questions were formulated. Main aim was to limit scope of this study to cost avoidance only as numbers of economic models are already present in literature [47] [19] [49] [51]

R.Q.1: What reuse economic models are present in literature which measure costs avoided through software reuse?

Research questions below are based on outcome of R.Q.1

R.Q.2: How many reuse cost related models calculate the cost avoided through reuse in a direct way?

R.Q.3: What assumptions are made by each of these models?

R.Q.4: What metric is required in order to apply each of these models?

R.Q.5: What are the limitations associated with each of these models?

The population in this study is managers seeking the financial benefits of software reuse. Intervention includes application of cost estimation models to estimate cost avoided through software reuse or deterrents caused by it. Research questions are not based on comparison. The outcome of study will be the
identification of flaws in current reuse cost estimation models (addressing cost avoidance) and suggestions in order to improve the models. We have not forced any restrictions on context and experimental design.

2.1 Generation of search strategy

Following steps were followed in order to generate the search strategy.

2.1.1 Search Terms Construction

A two phase search strategy was used in order to create the search terms. Both phases are combined in Fig 2.1. In the first phase following steps were followed in order to construct the search terms used for the study.

- Develop the search terms from research questions. All possible terms which relate to the research questions were listed.
- From the above list of terms, the synonyms of terms were added in list of search terms. Intention was to add as many search terms as possible.
- New search terms were created by changing the plurals to singular forms and singular to plural forms.
- By gathering keywords from abstracts and conclusions of relevant research papers.
- New search terms were gathered by browsing through grey literature (technical reports, articles etc.).
- By using Boolean OR with synonyms of search keywords.
- By using Boolean AND for combining different search terms.

In second phase search terms were piloted on an IEEE database. Each term was queried on database, based on the results obtained from the query few search terms were removed. This piloting of search string helped in improving the quality of search terms. These terms, used with different combinations, are listed in Table 2.1.

2.1.2 Electronic Database

Electronic databases are useful resource to find out relevant literature. Several different databases are present online with their own area of specialization. This study was concerned with software engineering therefore it was decided to select relevant databases only. Following databases were selected to perform this study:
Search strings were searched in the document titles. Motivation behind this choice was to simplify our search results and in order to get as many relevant results as possible.
Chapter 2. Related Work (Systematic Literature Review)

<table>
<thead>
<tr>
<th>No.</th>
<th>Search Term</th>
<th>No.</th>
<th>Search Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Software</td>
<td>16</td>
<td>return</td>
</tr>
<tr>
<td>2</td>
<td>Reuse</td>
<td>17</td>
<td>investment</td>
</tr>
<tr>
<td>3</td>
<td>Cost</td>
<td>18</td>
<td>Finance</td>
</tr>
<tr>
<td>4</td>
<td>Model</td>
<td>19</td>
<td>Assessment</td>
</tr>
<tr>
<td>5</td>
<td>Models</td>
<td>20</td>
<td>Business</td>
</tr>
<tr>
<td>6</td>
<td>Measurements</td>
<td>21</td>
<td>Reusability</td>
</tr>
<tr>
<td>7</td>
<td>measuring</td>
<td>22</td>
<td>measurement</td>
</tr>
<tr>
<td>8</td>
<td>framework</td>
<td>23</td>
<td>technique</td>
</tr>
<tr>
<td>9</td>
<td>estimation</td>
<td>24</td>
<td>reduction</td>
</tr>
<tr>
<td>10</td>
<td>Savings</td>
<td>25</td>
<td>price</td>
</tr>
<tr>
<td>11</td>
<td>Measuring</td>
<td>26</td>
<td>Tracking</td>
</tr>
<tr>
<td>12</td>
<td>benefits</td>
<td>27</td>
<td>Monitoring</td>
</tr>
<tr>
<td>13</td>
<td>Economics</td>
<td>28</td>
<td>Cost-benefit</td>
</tr>
<tr>
<td>14</td>
<td>Benefits</td>
<td>29</td>
<td>Financial</td>
</tr>
<tr>
<td>15</td>
<td>Avoidance</td>
<td>30</td>
<td>ROI</td>
</tr>
</tbody>
</table>

Table 2.1: Search Terms

2.2 Study selection criteria

Reuse economic models have been focus of researchers for over two decades and variety of models are already present in literature. Frakes [17] and Poulin [53] categorize the reuse economic models into different categorize. These categorize are based on the solution provided by each of reuse economic model. In order to limit the scope of this study, a reuse economic model category called Cost Avoidance [53] was selected as main focus of this research. Criterion was formulated and this criterion is further divided into exclusion criteria and inclusion criteria.

Inclusion criterion describes the rules by which articles are included in the study. An exclusion criterion was developed in order to exclude the irrelevant studies in the initial phase.

2.2.1 Exclusion Criteria

These guidelines were followed while excluding studies from initial search results.

- Studies which do not relate to software engineering
- Studies which do not relate to Software reuse.
- Studies which do not relate to cost aspects of software reuse.
- If full text of the study was not available.
• Exclude the studies which are in language other than English and English translation is not available.

2.2.2 Inclusion Criteria

In order to increase the credibility of study, inclusion criterion was made very flexible. It was made to accommodate every type of reuse economic model. Motivation was to search for models which can in-directly calculate cost avoidance. Following are the rules developed for including a study.

• Study discusses cost related aspects of software reuse.
• Study describes any software reuse cost model.
• Study should be available in full text.
• Study should discuss cost avoidance related topics
• Study should be published in some reputable journal and peer reviewed.
• Study describes validation of any reuse cost estimation model.
• Study evaluates or analyzes reuse cost estimation process or method.
• Study compares or discusses attributes of any reuse cost estimation model.
• Literature review, systematic review, industrial experience, case study, experience report on reuse cost estimation models will be included as well.

2.3 Quality Assessment Criteria

<table>
<thead>
<tr>
<th>No.</th>
<th>Quality Assessment Checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Does the article clearly define the context of study?</td>
</tr>
<tr>
<td>2</td>
<td>Is the introduction and background provided in detail?</td>
</tr>
<tr>
<td>3</td>
<td>Is this article cited by other articles?</td>
</tr>
<tr>
<td>4</td>
<td>Is the research methodology defined in detail?</td>
</tr>
<tr>
<td>5</td>
<td>Are the findings closely related to each other?</td>
</tr>
<tr>
<td>6</td>
<td>Do the objectives of research and conclusion relate to each other?</td>
</tr>
<tr>
<td>7</td>
<td>Is this research useful for Software engineering industry?</td>
</tr>
<tr>
<td>8</td>
<td>Does the article report the negative findings?</td>
</tr>
</tbody>
</table>

*Table 2.2: Quality Criteria*

To judge the quality of the included studies, a quality assessment criterion was developed. Quality criterion will help the study during analysis and synthesis
phase. It will be easy to differentiate between a quality study and an average study. Furthermore it will provide justification for providing more importance to selected studies based on its higher quality as recommended by [35]. This criterion is shown in Table 2.2.

Table 2.2 is based on check list and each field will be answered with yes or no. However, this quality criterion is not in very rigid and detailed. This is an exploratory study where objective is to locate all the economic models which can measure cost avoidance. This study was not designed to judge the quality of reuse economic models present in the literature.

2.4 Data Extraction

Dyba and Dingsoyr [14] have described procedure for the management of citations. This procedure [14] was followed during this study. Endnote was used for storing the citations in the initial phases of study. Separate endnote library was maintained for each step of the study. During the last stages of study an excel sheet was maintained in parallel with Endnote citations. That sheet was used for applying inclusion and exclusion criteria for the purpose of narrowing down the study results. Endnote was also used to find the duplicate studies and remove these studies in the initial phase. A data collection form was developed for extracting the information from the selected studies. A pilot study was conducted in order to check the usefulness of the data collection form, based on the input received, from the pilot study, this form was modified slightly. Data extraction form is present in Table 2.3

2.5 Conducting the review

After defining the review objectives and review protocol next step was to conduct the review. Conducting the review was several step processes. The conducting of review is shown in figure 2.2. In the first stage, each database was searched for relevant studies. All the defined search terms were used one by one and results were collected in Endnote. In order to get maximum results all results obtained were included in this phase. These seven databases generated 9348 results. Several copies of each article were present in the initial search results. Table 2.4 shows results obtained from each individual database.

In second stage duplicate entries were removed. There were multiple copies of articles. This removal was a two step process. In the first step articles were removed based on exactly similar titles. Endnote was used for that purpose. In the next step articles were sorted with respect to authors name and then manual reading of titles took place. In total 7388 articles were removed from the initial search result.
Chapter 2. Related Work (Systematic Literature Review)

- **General Information**
  1. Date of data extraction
  2. Article title
  3. Authors Names
  4. No of references
  5. Journal/conference/conference proceedings
  6. Date of publication

- **Specific information**
  7. Study context (Academia or Industry)
  8. Research methodology Literature review / Systematic review /
  - Case study/Experiment / Survey/ Action research
  9. Study subjects Professional / Student

- **Research Questions**
  10. Name of the presented model
  11. How was the model presented Equation / diagrams / description
  12. On what grounds model is constructed
  13. What were the aims of the article?
  14. Is this model an extended form of any other model?
  15. Can the model calculate costs avoided through quality improvements?
  16. What metrics need to be collected in order to apply the model?
  17. Does the model provide guidelines on how to collect the metrics?
  18. What is the basic formula used by this model?
  19. Can the model calculate costs avoided through artifacts other than software code?
  20. How can model calculate costs avoided through artifacts other than code reuse?
  21. What different types of software artifacts were considered in the article?
  22. Was the model validated in Industry?
  23. Was the model validated in academia?
  24. Does the model provide details of validation in industry?
  25. Is the model dependent on organizational, technical or managerial processes in the industry?
  26. Is the model dependent on systematic reuse processes?
  27. Any assumption which model makes?

*Table 2.3: Data Extraction Form*
Third stage involved application of exclusion and inclusion criteria. First the exclusion criterion was applied in order to eliminate irrelevant studies so that remaining studies were only on software reuse. Inclusion criteria were then applied in order to narrow down the studies to economic models of software reuse. Generic studies on various topics of software reuse were eliminated. Only title was considered while applying inclusion, exclusion criteria. However in some studies it was difficult to judge the content of study based on its title, these studies were included for further review.

Abstracts are good way to describe an article. Abstracts from 497 studies were reviewed. After reading the abstracts 77 studies were shortlisted for further investigation. These studies either contain reuse economic model or their abstract was inconclusive and demanded further review of studies. Conclusions of these 77 studies were reviewed. After reviewing the conclusion 29 reuse economic models

---

**Figure 2.2:** Multi-step filtering of studies and final number of primary studies
were found. These 29 studies are present in Appendix B. Full Text of 15 studies was not available. These studies were eliminated from the review. List of these studies is present in Appendix C.

Objective of this review was to find reuse economic models which can measure cost avoided through reuse. The focus was not on general reuse economic models. Therefore these 29 studies were manually reviewed by reading their content. Finally 3 models were found which directly address the cost avoided through software reuse. These three studies are listed in the table 2.5. These articles are focused on cost avoidance and provide mechanism to measure the costs avoided through reuse.

In review some models were categorized as cost benefit model or return on investment models but these models were flexible enough to calculate cost avoidance as well. Measuring cost avoidance was not a focus of such models but they were generic enough to accommodate cost avoidance measurements. Since these models were not made for measuring cost avoidance therefore these were excluded from this study. These studies are listed in table 2.6.

<table>
<thead>
<tr>
<th>No</th>
<th>Ref</th>
<th>Article Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[53]</td>
<td>A reuse measurement and Return on Investment Model</td>
<td>1993</td>
</tr>
<tr>
<td>2</td>
<td>[12]</td>
<td>Software Reuse Metric Plan</td>
<td>1993</td>
</tr>
<tr>
<td>3</td>
<td>[1]</td>
<td>Measuring benefits of Software Reuse; Examining three different approaches to software reuse.</td>
<td>2005</td>
</tr>
</tbody>
</table>

*Table 2.5: Final Models*
Chapter 2. Related Work (Systematic Literature Review)

<table>
<thead>
<tr>
<th>No</th>
<th>Ref</th>
<th>Article Title</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[42]</td>
<td>Economics of software reuse revisited</td>
<td>1993</td>
</tr>
<tr>
<td>3</td>
<td>[50]</td>
<td>Cost-benefit analysis for software reuse A decision procedure</td>
<td>1994</td>
</tr>
<tr>
<td>4</td>
<td>[57]</td>
<td>Cost Estimation Model for reuse based software products</td>
<td>2008</td>
</tr>
<tr>
<td>5</td>
<td>[65]</td>
<td>Evaluating software reuse alternatives A model and its application to an industrial case study</td>
<td>2004</td>
</tr>
</tbody>
</table>

*Table 2.6: Models which are not developed for Cost Avoidance*

2.6 Analysis and Discussion

After conducting the review and collecting the data, next phase was to focus on the analysis of models. Limited numbers of models were developed for calculating cost avoidance. Models which calculate the cost avoidance indirectly and were not made for measuring cost avoidance were excluded in this analysis phase. These models are listed in table 2.6. Reason for eliminating these models was simple, this study focus only on the models which are made for calculating cost avoidance. Any model which was not made for calculating cost avoidance was removed from the analysis and discussion part. In this section some main characteristic of models are highlighted and effect of these characteristics is discussed.

2.6.1 Available Models for Cost Avoidance:

Following research questions deal with this section.

R.Q.1: What reuse economic models are present in literature which can measure costs avoided through software reuse?

R.Q.2: How many reuse cost related models calculate the cost avoided through reuse in a direct way?

In this section a brief description of these three models is provided

**Poullin and Carusos Model**

Model [53] proposed by Poullin and Caruso is applicable on individual project or individual team. This model is divided into two phases. One phase in concerned with costs avoided during development and the other phase focus on costs avoided during maintenance. According to this model costs avoided during maintenance can be measured by following equation:


\[ DCA = RSI \times (1 - RCR) \times \text{New Code Cost} \]

Here,

DCA = Development Cost Avoided

RSI = the total lines not written but included in the source files. RSI includes only completely unmodified reused software units [53].

RCR = Relative cost of Reuse i.e. 0.2

This model assumes that users will be aware of cost per lines of code. Model is based on lines of code therefore no other software artifact can be measured with this model. Maintenance costs avoided by this model are termed as service costs. These costs take into consideration error rates and number of errors occurred after the new artifact has been in use.

**Defense Information Systems Agency**

United States department of defense information systems proposed a model similar to Poulins model [12]. In fact this model is a more complex form of Poulins model. This model measured the total effort in labor hours to develop a project with reuse. This is shown in following equation:

\[ CA = R \times (En - Er) - CCOTS \]

Where:

CA = Cost Avoided

En = Estimated Effort without reuse

Er = Estimated Effort with reuse

CCOTS = Commercial Off the shelf Component

R = Costs of New custom code (in person hours)

It has similar limitations as previous model. This model is also based on lines of code and do not take into consideration any other forms of software reuse.

**Measuring the benefits of software reuse - Examining three different approaches to software reuse**

This is new and comprehensive models on cost avoidance through software reuse [1]. In this model authors, Amar and Coffey, provided three different models for measuring reuse savings. They take into consideration the processes present in the organization (ad-hoc or systematic) and based on the processes any of the provided formula is used. Basic unit for this model is person-hour.

Authors proposed the basic formula as:

\[ (TLR+U) \times N + i \times SR \times MOD + i \times (1-SR) \times BUILD \pm i \times BUILD \]
After simplifying for ad-hoc reuse,

\[
\text{percent savings} = \left[ \frac{\text{SR} - (\text{TLR} + \text{U}) \times (N/i)}{\text{BUILD} \times \text{SR} \times (\text{MOD} / \text{BUILD})} \right] \times 100
\]

\( \text{TLR} \) = Time to locate each potentially reusable item  
\( \text{U} \) = Time to understand suitability of each potentially reusable item for current task  
\( N \) = Number of items that were examined, including each of the items that finally get reused  
\( i \) = number of attempted instances of reuse  
\( \text{SR} \) = Search hit rate. Percentage of \( i \) that yielded a positive search result  
\( \text{MOD} \) = Time to integrate/modify the reused item for current purposes  
\( \text{BUILD} \) = Time to build an element from scratch

Here they assumed that organizations collect metrics such as search hit rate, time to understand suitability etc. Secondly this method provides the value of search hit rate as 20 percent but failed to explain the rationale or any reason behind this decision. This limits the models application as most of the organizations are not interested in collecting complex metrics (like search hit rate, time to integrate etc) and it can be consider as an overhead for the organization. This model lacked clarity and necessary details such as how to collect data, what should be counted as reuse are missing. Focus of the article is to provide model for cost avoidance and validating in the industry is not discussed.

### 2.6.2 Metrics required

This section deals with the following research question: R.Q.4: What metric is required in order to apply each of these models?

Two models were based on the lines of code [53] [12]. In order to apply these models, one has to know the number of lines of codes reused and total size of the product in terms of lines of code. Third model [1] was based on person-hour as a basic metric. This model was based on several other factors such as search hit rate and number of attempted instance of reuse. Table 2.7 shows the metric used in each of the short-listed model

<table>
<thead>
<tr>
<th>No</th>
<th>Model</th>
<th>Metric Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[53]</td>
<td>Lines of Code</td>
</tr>
<tr>
<td>2</td>
<td>[12]</td>
<td>Lines of Code</td>
</tr>
<tr>
<td>3</td>
<td>[1]</td>
<td>Person-hour</td>
</tr>
</tbody>
</table>

*Table 2.7: Metrics Used*
2.6.3 Limitations of Models

Research questions 3 and 5 deals with this section.

R.Q.3: What assumptions are made by each of these models?
R.Q.5: What are the limitations associated with each of these models?

Reuse artifacts, present in literature, are of various different types [36]. First thing to consider about the limitation of any model was to investigate kind of artifact it can measure. Table 2.8 shows artifacts and the models [1] [53] [12] and checks that if models can measure these software artifacts. Yes or No answers are made when artifacts can be measured or cannot be measured. However, in some cases it was difficult to know whether the artifact can be measured or not. In such cases May Be was used and further investigation is required.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>No</td>
<td>No</td>
<td>May Be</td>
</tr>
<tr>
<td>Architecture</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Data</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Design</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Documentation</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Estimates</td>
<td>No</td>
<td>No</td>
<td>May Be</td>
</tr>
<tr>
<td>Human Interfaces</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Knowledge</td>
<td>No</td>
<td>No</td>
<td>May Be</td>
</tr>
<tr>
<td>Models</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Plans</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Requirements</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Service Contracts</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Cases</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Code</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Table 2.8: Artifacts measured by Models*

2.6.4 Validation in Industry

Software reuse yields its benefits for the industry and all studies related to software reuse should preferably be validated in industry. By validating a study in the industry researchers can find out the limitations and advantages of their studies. Final studies were checked for their validation in the industry. Table 2.9 describes the validation aspect of the short listed models.
Chapter 2. Related Work (Systematic Literature Review)

### Table 2.9: Validation of models in industry

<table>
<thead>
<tr>
<th>No</th>
<th>Model</th>
<th>Metric Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[53]</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>[12]</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>[1]</td>
<td>No</td>
</tr>
</tbody>
</table>

#### 2.6.5 Quality Assessment

Quality assessment allows us to judge the quality of study. Quality criteria were developed and selected studies were passed through these criteria. Table 2.10 shows the results after applying the quality criteria. It was interesting to note that no study reported any negative findings from their proposed model.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the article clearly define the context of study?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Is the introduction and background provided in detail?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Is this article cited by other articles?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Is the research methodology defined in detail?</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Are the findings closely related to each other?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Do the objectives of research and conclusion relate to each other?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Is this research useful for Software engineering industry?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Does the article report any negative findings?</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**Table 2.10: Quality Assessment Checklist**

#### 2.7 Conclusions and Discussion

In previous section discussion on various findings of this systematic literature review is present. Following conclusions were drawn from this study:

**Conclusion 1:** Lines of code, as metrics for calculating cost avoidance, cannot measure different types of reuse artifacts. Therefore, any other metric should be introduced in order to measure all kinds of reuse artifacts.

Most common reuse artifact was the code. Nearly all models are based consider it as an only artifact which can be reused therefore these models are based on lines of code. However, a study by Bhargav [36] listed 14 different artifacts which can be reused. This study was based on artifacts which are present in literature. Lines of code cannot be a measure which could be applied on any reused
artifact. Hence, all these models have questions marks on their applicability on artifacts other than code.

**Conclusion 2:** Basic formula, for measuring cost avoided through software reuse, remains the same in all models i.e. Cost without Reuse Cost with reuse.

All models calculate reuse cost avoidance by same basic formula. However, metrics used in the formula differs in models. Most models used lines of code as a basic measure while one article used person-hour as basic measure for calculating software reuse. No time value of costs was included, which is usually the case with cost-benefit reuse models. Formula in all the models was sub-divided into integration, testing, searching and other costs related to reuse. All these costs sum up as a cost with reuse. Cost without reuse was generally calculated from historical data.

**Conclusion 3:** New models are not validated in industry yet. Literature contains validation experiences of older models only.

The most recent model on cost avoidance was not validated in industry. No experience of validation was presented in the article. However, old models, based on lines of code, were validated in the industry. They have been used in the industry for number of years.

**Conclusion 4:** No negative findings were reported by the models. None of the models described any negative findings related to it. Literature lacks any articles which reports negative experience of these models. Negative findings help the practitioners in evaluating a model before its usage. Therefore its a major drawback associated with these models.

**Conclusion 5:** No guidelines on what to collect and how to collect metrics related to model. Collection of metrics can cause problems for industry practitioners. Guidelines should be provided on what to collect and how to collect. None of the models described these guidelines.

### 2.8 Limitations

There were studies of which no full text was available. These studies were very old and hard to locate. This has been a problem which can affect the validity of study. [30] has already reported this problem in literature and its affect. These studies were not used in this literature review since it was not possible to locate these. Therefore its a major drawback associated with this review. These studies can contain useful information on cost avoidance.

Research focused on models which only deal with cost avoidance. There are models which can calculate return on investment, cost-benefit analysis and cost avoidance within a single solution. However these models are complex to apply and metrics are hard to collect. Therefore these were skipped in this study. However, list of these models is available in Appendix c.
2.9 Summary of Literature Review

Cost avoidance is categorized as a category of software reuse economic models. However, in literature very little information is present on this topic. Aim of this study was to investigate about all the economic models built specifically for calculating cost avoidance through reuse. This study was conducted by Kitchmans method of systematic literature review. Results show that there are only three models which are developed for calculating software reuse. Two models are based on lines of code as a basic metric while third model is based on person-hour as a basic unit for measuring software reuse. Study found some limitations related to these models and these are listed in previous sections. Major problems found in these models are related to their application for industry, complexity level and lack of guidelines on collecting software metrics. Limitations are mentioned in Section 2.8, the main limitation was the lack of full text available for some articles.
Chapter 3

Background Work

This chapter contains description of background work done on this thesis. Three different studies are described and their details are narrated. Conclusion section of each chapter describes final outcome of each study.

3.1 A Survey on an organizations reuse process: Results and Analysis

A brief study was conducted on software reuse practices and benefits associated with reusable artifacts. In this study a web-survey was performed inside an organizations products. This survey was sent to the designers in the form of a web-link, where they have to fill out the questionnaire.

In the sections below author has gathered the results and analyzed the results. These results were formed on the basis of the feedback received from 6 designers. These designers had experience of developing large telecommunication systems.

This study focused on two aspects of reuse. One was related to the patterns and general trends of reuse practices inside organization and second was cost aspects of software reuse. This was an exploratory study and aim was to find out problems faced by the designers for reusing a software artifact.

3.1.1 Research objectives and methodology

This study describes basic trends and patterns present of reuse practices inside an organization. These trends and patterns can be helpful for the future research inside an organization and may point out the discrepancies present in the reuse process.

Main research objectives of this study were:

- To find the patterns of reuse and level of reuse inside organizational unit.
- To measure savings because of software reuse, identify method used by designers and problems associated with that.
Research method used for conducting this particular study was a survey. This method was selected because it was easier to reach the designers at remote location with the help of this method. A web survey was accessible by everyone on the intranet. Moreover, this study was meant to be a starting point for future research on same topic. In order to get the basic ideas of activities taking place inside an organization, a survey was quick and easiest method. This is categorized as qualitative research method.

Reuse activities are more concerned with the staff present at grass root level of software projects. Software designers were selected as the focal point for this survey. Major development and management activities related to the software reuse take place at the design level [64]. Therefore, for this survey the only focus were designers of different products.

Following steps were followed to perform this study:

- Identify the areas related to reuse which needs to be investigated. These areas were patterns of reuse activities, nature of reuse products and costs saved because of reuse.
- Questions were developed
- Questions were sent to the domain experts for review and feedback.
- Questionnaire was updated with the changes, based on the feedback received from the experts
- Request to the managers about the intended audience of survey questionnaire
- Publish the survey over a web pool, which is accessible for audience in focus.
- Receive results and analyze them.
- Compare the results with the literature and come up with the suggestions.

During analysis phase, results were divided into sections. Each section describes a particular research question.

3.1.2 Questionnaire

Designing the questionnaire, which can help to find the answer of research questions, was an important part. Questions related to the first research questions were influenced by the following literature studies [21] [28] [27] [45]. These studies helped to figure out what sort of questions needs to be asked from the designers. Questions were also designed keeping in view the different types of artifacts which can be reused by designers. These questions not only focused on code reuse but they were also directed towards reuse process itself.
Second research objective demanded numbers which can illustrate the cost savings because of reused products. Designing questions for a study about costs was difficult because no previous knowledge and figures were present inside the organization. Since this was an exploratory study, from which other studies can benefit, therefore questions were developed to understand the awareness level of designers about cost savings. These questions were mostly about identification of process used to calculate cost savings.

After preparing the questionnaire it was sent to the researchers for review. Basic aim was to make this study more useful and to maximize the benefits of this limited study. After getting the feedback from three active researchers, questionnaire was updated with the changes suggested by them.

Questions prepared were mixture of close ended and open ended questions. Close ended questions were mostly related to the patterns of reuse activities carried out by designers. Close ended questions were in form of selecting the best choice out of given choices. Open ended questions were aimed at finding the reasons behind any particular pattern. Some of open ended questions were left optional for the interviewee in order to limit the time for filling out survey.

Questionnaire was divided into two sections. One section was focusing on reuse patterns, level of reuse inside the products and awareness about effectiveness of reuse process. This section comprises major part of questionnaire. Second section was related to cost benefits associated with reuse process. Basic aim was to find costs saved by reuse process. However, since it was expected to be very hard for the designers to quote any figures therefore main focus was to find the process of calculating cost savings inside organization products.

### 3.1.3 Data Collection

Collection of data was done using the organizations internal web tool. Survey was published on web resource. All the designers received an email with a link to that resource. They can fill there choices and write there comments on asked questions. These comments were then stored and managed by a web based tool. Owner of the survey was able to see results in different forms. A basic summary was shown in statistics form. However, if someone wants to see the detail results and which interviewee posted what kind of reply, further advanced options were also available. Data received from the interviewees could also be imported in an *.xls format and can be stored locally for analysis.

### 3.1.4 Results Obtained

Feedback received from the designers was enough to show the patterns of reuse practices.

Results obtained from the survey are present in this section. These are listed
under two main categories, pattern of reuse and cost factor. Each result is elaborated in the respective paragraph.

3.1.5 Patterns of Reuse

Identified patterns of reuse are discussed under their respective headings

Reuse Culture

From the feedback received from designers it can be concluded that reuse culture is ad-hoc. Normally designers try to look for reusable artifacts while designing a new product but often end up developing their own because of various issues. Efforts of reuse are at individual level and do not have proper procedure present which can assist the designers in searching for the required component. According to literature such a reuse culture is called opportunistic reuse [26]. Searching for proper component is part and parcel of the reuse process. It often requires a lot of effort. All designers complained about this issue. Such complaints showed the lack of facilities at organizational level

Open-source components are popular

When asked about the favorite artifact, nearly all the designers preferred Open-source components for reuse. Reason behind this unanimous choice can be the support available for open-source components. Normally, OSS has wikis and forums available for the support. It has experts available who are willing to help the user if he has some queries related to the reuse. Access to these forums is easy and queries posted on these forums receive a quick reply most of the time. Therefore most designers prefer OSS. Literature also points out the success of OSS when reused[48].

Middleware is preferred inside product family

Nearly all designers used middleware as a preferred reuse artifact. Here at An organization most of telecom products run as a service in application servers. They can be labeled as Domain Specific applications [53]. In literature these are classified as most used artifacts in a code reuse [53]. They fall under the category of middleware. These are easily reused several times without any major changes and that can be the reason behind the popularity of middleware components. Secondly, developing a middleware component from scratch is a troublesome job; therefore most of the designers like to reuse an old one.
GUI is least preferable

Surprisingly GUI components are least likely by the designers to reuse. Designers think that they had to modify a lot of code in order to reuse any GUI component. Possible reason behind this is that the requirements are different for each GUI therefore one standard GUI cannot be reused several time. However, middleware and other artifacts most of the time remain same throughout the domain.

Problems because of lack of owner-ship

While describing the problems related to reuse, nearly all the designers mentioned lack of support present for the reusable artifacts. Designers do not know whom to contact when they want to inquire about any problem in reusable artifact. There is no central owner present who can answer the queries related to the issues. Literature is full of experiences where owner-shop of artifact caused revolutionary changes reuse process. \[27\] \[46\] \[24\]

Test cases are favorite artifact other than code

Ease with test cases are understood and modified, made them the favorite artifact of designers, other than code. Normally, test cases are in human understandable language and require fewer changes in order to reuse them. Therefore, designers interest is justifiable.

Multiple factor affecting reuse

Since reuse activities inside an organization are ah-hoc based therefore every designer faced different problem than other. Some reported problems concerning the lack of support while some reported about badly written components. No single factor was pointed out which was affecting the reuse activities inside an organization. Therefore, multiple factors are affecting the reuse activities.

No central repository is present

No central repository was reported by the designers. Each designer named different software applications to store any component for future reuse. Repositories used by the designers were locally accessible for their developers however they were not accessible for rest of organization employees.

Modification of Classes

Most of the designers reported that they do not have to modify large parts of artifacts in order to reuse. Modification in less than 15 percent of classes was needed in order to integrate the reusable artifact. These values were estimated by the designers in a reply to a question.
This means not much effort was wasted on tweaking and modifying the components, which is good thing for reuse. Similarly this shows lack of black box reuse which is recommended practice in literature.

No experience of sharing own component

None of the designers shared any experience about their own component which was reused. When asked about the possible ways they can provide support for their own components everyone considered Email as easiest solution. However, author thinks that multiple different ways should be used to provide support for the reuse artifacts. Some of them are wikis, documentation and product owner. This practice is also reported in literature [46].

![Figure 3.1: Sharing own component](image1)

Ideal reusable artifact

When asked about the property which makes an artifact as an ideal reusable artifact, majority of the designers were of the view that easy to use and well defined interfaces should be present in the artifact. This reduces the time for understanding an artifact and makes reuse easy for designers.

![Figure 3.2: Ideal reusable artifact](image2)
3.1.6 Cost Factor

Study about cost saving possibilities showed interesting result. Since this study was made at designers level, initial expectations were that designers know about the cost benefits. However, data received from the survey pointed out that designers have no knowledge about cost factors. They do not consider cost savings as a major reason for reuse.

When asked about their reason behind reusing any artifact, they pointed out short development time as main incentive. Although this shorter development time indirectly reduces cost but designers should be aware of the costs related to the reuse itself. This will further support the reuse activities and will help in making the decisions related to reuse. Designers can see two dimensional aspects, one related to shorter development time and one related to cost savings. In decisions where more development time but less costs (because of lesser resources required) are involved then designers can get the benefit out of this two dimensional approach. An example of such scenarios can be an Research project which sometime has limited resources assigned to it but have not clear end time.

Second interesting finding was lack of any procedure in order to calculate cost benefits of reusable component. Designers do not know about any procedure through which they can judge the feasibility of any reusable artifact.

![Figure 3.3: Knowledge about Cost Estimation](image)

Presence of a proper procedure, to estimate the cost savings and development time saved by reusing any artifact, is necessary thing for reuse activities. In literature different cost models like COCOMO and its variants are present. Research articles also described different cost models to estimate the costs saved and financial aspects of reuse [53] [49] [51]. These can be used by the organization in order to get the maximum benefits from reuse and keeping track of costs. No organizations can be fully sure of the benefits it is getting from reuse unless there are metrics and measures present for calculating costs and savings. By having costs in a documented form it will be easy for management to set goals and analyze the productivity of the organization.
Chapter 3. Background Work

3.1.7 Interpretation and analysis of results

Patterns of reuse depicted in the survey feedback indicate that reuse activities are taking place at individual level. Designers tend to think about reuse activities and how can they reuse any artifact in a new release or new product. Awareness and intent is present but lack of guidance and procedure often make it hard for them to reuse any artifact. Such a reuse is known as opportunistic reuse [24]. This is a most basic form of software reuse and does not yield proper benefits for the organization. In such a reuse procedure, chances of producing quality product in a shorter time development time are rare. Usually the organizations aim for a systematic reuse process [24].

Open Source software artifacts were considered as favorites by the designers. This indicates a lack of support or lack of internal artifacts present for reuse. Concept of a software reuse is a two dimensional concept [48]. One dimension deals with actual reuse of the artifact while the other dimension is for producing a reusable artifact for the future reuse. By indicating there interests in open source software artifacts designers have pointed out two major problems present inside the organization. Firstly, there are either no internal producers of reusable artifact or there is no support provided by them. This analysis is confirmed by another interview question when asked about characteristic of an ideal reusable artifact some of the designers were interested in availability of product owner. In some of other questions when asked about sharing experience with us about their own reusable artifact, designers fail to describe any of their experience. This indicates that producers of reuse artifacts are not present. Lack of systematic reuse was also evident when designers complained about multiple factors which affect reuse. They were unable to list a single common factor which makes life tough for the consumer of reusable artifact. Factors varied from the management related activities to technical activities. Therefore; we can conclude that reuse activities can flourish with presence of product owner and by following a proper procedure for producing a reusable artifact [46].

Feedback regarding modification of code indicates that White Box [64] reuse is the general trend inside the organization. There are no ready to use artifacts which are present in the organization, as recommended by the literature. [24] [46]. White Box reuse is often troublesome and do not guarantee a shorter development time and low costs. Even after integrating the reusable artifact by the means of white box reuse, the product needs rigorous testing. Chances of introducing a defect in a reusable artifact increase with the white box reuse[55]. However, Black box reuse is easy to introduce in a new product and guarantees a quality product in a shorter development time. Black box reuse is the standard practice present in the literature [55]. Black box reuse can only be possible when there is a systematic way of producing the reusable artifact internal to the organization. Otherwise, search for the third party reusable artifact takes a lot of resources and time.
An interesting finding was related to reuse of GUI components as the least preferable reusable artifact. All the designers think that GUI are there lest favorites. Reason behind this phenomenon can be the lack of standardize GUI components present. GUI components may vary from application to application and that makes it hard for the designers to reuse them in their own product. Usual expectations from this question were that the GUI components should be the favorite of designers since most of the organization products belong to the same application family. GUIs in these products use the standard look and feel therefore reuse should be very easy. But the results indicate that many other factors are also driving this design decision.

When asked about the favorite way of providing support for your own component most of the designers supported Email as there preferable tool. Since, email has become a standard medium for official communication therefore these results were not unexpected. However, making email the primary medium of communication can cause a problem. A product owner can become irritated of answering the same questions again and again. This can take his useful time for answering the already solved problems. Therefore, recommendation from the author is to use variety of different mediums of communication like forums, wikis and emails. Large organizations have found various ways to tackle this issue [20] [33] and an organization needs to find a way which is best suited for its structure.

Cost savings effort at the designers level demand an awareness about the cost related issues in the designers. This was lacking among them. They just think how they can save the development time. In actual case, a decision based on shorter development time may be misleading for the organization. Some projects require more time for the understanding part of the problem as compare to the solution part, in such cases it is difficult to find the actual development time. Therefore decisions bases solely on development time can cause problem for the management. Therefore, costs should be considered by the designers while reusing any artifact.

3.1.8 Recommendations

Making the reuse process institutionalized should be the prime aim of every organization. A reuse process can yield maximum benefits in a consistent way if a standard well defined reuse process is present and followed inside the organization [55]. This leads to a systematic reuse process as defined by the literature. This process can vary according to the needs of the organization. Therefore, a well defined process and guidelines for using that process are required. These two changes can be started by the management only. Effect of these two changes may not be evident immediately but once the reuse processes are well established positive results can be achieved. Frameworks in literature are present which can help an organization in obtaining a systematic reuse culture. [65]

Much emphasis of reuse inside the organization is on the consumption of
Chapter 3. Background Work

reuseable artifact. A lesser focus is on the production of a reusable artifact. A reusable process can benefit a lot if well defined reusable artifacts are produced internally according to the future needs of the organization [53]. By producing the customized artifacts ready for the reuse process, a lot of time and effort can be saved. This saving of time and effort improves the savings as well as improved the quality of the product itself. The concept of producing the reusable artifact is an old concept [53] which is not widely practiced in the industry. However, by following this concept an organization can save maximum amount of cost.

Designers which try to reuse any software artifact complained about the lack of support present for the artifact. This is a critical problem and no reuse activity can be successful without solving this problem. No product owner is present which can help the designers in case of any issues. The designers supported the use of open-source components because they can find support for their queries. Hence, reuse can be made integral part of the development process if each reusable component has some product owner. In literature we can find recommendations about the product owner [6] [21] [27] at several places.

Domain analysis is an important practice which takes place in organizations with large application product families. In this practice, components and artifacts are listed which can be shared among the different products. Further analysis is done on how to make the reuse of common artifacts easy and with less cost. This analysis should be done on the major organization products in order to list the common artifacts. In future releases and updates the listed reuse resources can be used to save the costs and time.

Reuse activities inside an organization can increase if a central repository is present where only very high quality artifacts are stored. This will save time for searching the new artifact and also improve the quality of the final product. Each component should have a product owner team. This team can comprise of number of resources from different levels of project development. A product owner should lead the team who should be responsible for solving the problems related to the reuse of the artifact after consultation with the team members. This may not be a dedicated team but used on occasional bases depending on the priority of the request Wikis and emails should be used for storing documentation and relevant information about artifacts. Black box reuse should be encouraged. Even for the components where code is involved, interfaces should be designed in such a way that they can support future reuse of that artifact.

Designers should be aware of cost benefit analysis of reusable component. Since an organization is aiming for cost reduction because of reusable components therefore at design level this awareness should be present.

3.1.9 Future work

Future work, based on this study, should be more focused on two different aspects, cost savings, and effect of systematic reuse on organization. These two are
described below.

Cost avoidance and financial benefits are major reason behind the usage of reusable artifact. However, organizations do not know what exactly is the amount of reuse present inside there development unit and how much costs have been saved. A framework should be developed which can help the organizations to find status of reuse and savings associated with it. These saving can be direct savings like not having to buy any artifact or these can be indirect savings like reducing the number of possible defects. A major study should be launched on this aspect.

Systematic reuse can improve process of reuse, making it easy for stakeholders to reuse artifacts. At the same time it is important not to forget the continuous improvement of this process. Therefore, a small study should be launched, after the implementation of systematic reuse process, which measures the effect of new guidelines over other operations inside the organization.

3.1.10 Validity

This study was of limited duration and scope. Major objectives were partially achieved because of lack of feedback received from designers. However, feedback received showed some important patterns but they can only be confirmed by a major study in future. The validity of study has a question mark on it since all the feedback received was compared with literature and then interpreted and narrated here. If some problems are not reported in literature then these cannot be identified with help of this study. In future studies, these two things can be fixed by adding more time for the study and by persuading the designers to answer the questions. Interviews can be best suited for such exploratory studies since interviewee can express his feelings in a better way. He can talk about things which researcher has ignored in his interview questions. In short this study has questions marks over it and its results need further validation.

3.1.11 Conclusion

In order to know the reuse activities present inside an organization a survey was conducted at designer level. This survey showed that reuse activities are ad-hoc based and these activities are present at individual level. No central procedure or guidelines are present inside the organization to support these activities. Therefore, each designer is free to use his own method and procedure for reusing any artifact. Most of designers want a product owner for any reusable artifact, who can answer their queries related to the component. Open source components were the most popular reusable components because they have enough support present if some questions arise. Designers were unable to answer cost related questions. They do not perform any cost-benefit analysis before reusing any component.
They opt for reuse only because it can reduce development time. Recommendations are also provided by the author in order to move the reuse culture inside an organization from Opportunistic reuse to Systematic reuse. Suggestions like domain analysis, institutionalization of reuse process and many small changes were listed in order to improve reuse process. Recommendations for future work were to launch one study on cost benefits and one separate study on the effect of systematic reuse over organizations processes. Overall this study proved to be a good exploratory study on the reuse activates taking place inside an organization. It can help the organization in understanding its process and in launching future research on the reuse.

### 3.2 Literature Survey on Reuse Practices of 5 Large organizations

This research study tried to identify various success factors which are considered important by various organizations. A review of literature pointed out industrial experience shared by 5 different organizations. Details of experiences are present in section 2. Section 3 contains analysis and discussion regarding the results. Section 4 contains recommendations for introducing reuse based development program in an organization.

#### 3.2.1 Industrial Review

In this section a review of reuse practices followed by 5 different companies is described. Each organization and its factors are described in respective subsections.

**Hewlett Packard (HP):**

Hewlett Packard started its focusing on reuse activities during mid-80s. From mid-80s to early 90s lot of investment had been done on making reuse an integral part of development process. Research on many aspects of software reuse was carried out. This research was driven by the problems and issues experienced during the reuse of various products. During the early 90s HP launched a comprehensive program on domain specific reuse. Here in this section, results gathered from two HP related studies, on reuse, are present. Various factors, related to software reuse, are presented as following. These factors are identified from articles [23] [40].

**Motivation**

Main motivation behind HP’s program was to reduce time to market by reusing common features present in various products.
Technology Transfer Method:
HP developed its own method for transferring new technology of software reuse. Details about this method are missing in literature. This method enabled HP developers community to learn new technologies and practices related to software reuse.

Non-Technical vs Technical Problems
According to articles published on reuse, HP reuse community feels that non-technical problems are major hindrance in reuse related problems. These problems vary from searching / locating of reusable item to managements non-commitment to reuse program. Technical problems were considered to be less important and solvable by the reuse community.

Management Support:
According to articles, researchers feel that managements support for reuse program is a major problem related to software reuse. If management is not willing to support reuse initiatives then reuse related success is not possible in the organization. Designers or developers at low level could reuse artifacts on their own but organizations get major benefits only when software reuse becomes a priority in the organization.

Integration of Process, People and Technology:
Integration and alignment of processes, people and technology is important to achieve positives results related to software reuse. All these should be integrated in such a way that they have a common goal, objectives and aims. Working methodology to support reuse activities should be introduced and stakeholders should feel comfortable with it. Similarly technology which can support reuse of artifacts should be introduced; people should be trained for that.

Design for reuse:
HP research community believes that reuse cannot be successful unless component / artifact are built keeping in mind its future reuse aspects. For example, modern technologies (at that time) such as OOP and high level programming languages makes it easy for architects to design applications which are more coherent and less coupled with other applications. Such applications can be easily reused.

Tailored Reuse Methods:
Research articles describe the importance of tailored methodology for reusing artifacts. Each organization should develop is own methods, processes and guidelines for introducing reuse culture inside organization. These methods should be based on organizations ways of working and competence level present inside the organization. Tailored method will help in reducing possible non-technical prob-
lems related to reuse. Similarly organizations would not need any extra efforts for introducing reuse culture in the organization.

**Incremental Strategy:**
HP followed incremental approach for introducing reuse culture in organization. Changes were introduced gradually and no large scale changes were introduced immediately. Slowly but gradually organizational structure was molded to form a structure supportive of software reuse. Such strategy was acceptable for all stakeholders since they did not have to learn new things immediately. Usually, large scale changes were distributed over 5 to 7 years.

**Producer-Consumer Model:**
HPs reuse community developed its own producer-consumer model for reusing artifacts. This model was developed over several years and guidelines were developed in order to introduce reuse culture inside the organization.

**Customized Metrics and Cost Model:**
Another important factor in HPs successful reuse program was its measurements related to software reuse. Based on organizational structure HP developed its own customized metrics and cost models. These metrics and cost models kept track of progress made by the organization and savings because of reuse. This gives a good idea of success / unsuccessful reuse program. Decision making was done based on the results received from these models.

**International Business Machines (IBM)**
IBM is one of the famous organizations in the world which develops variety of software applications. Software reuse had been focus of IBM since early 90s. However, very limited amount of work is published by the organizations. Since, IBM has several similar style applications, reuse initiatives were taken in order to benefit form common reusable features. These applications vary from large scale development tools (Clear case etc) to small plug-ins (for eclipse etc). These factors are identified from articles [54] [68] . Success factors reported are described below:

**Motivation:**
Motivation behind IBMs reuse program was to improve quality of the product by developing high quality reusable artifacts and then reusing these artifacts in several products. Maintenance cost could be cut down in that way.

**Planned Reuse:**
IBM realized the importance of planned reuse culture inside the organization. Researchers inside the organizations believed that reuse can only be successful when planned at organizational level. Therefore, strategy was developed in order
to make reuse integral part. This strategy was based on the need of organization and several experts were responsible for developing this strategy.

**Design and Analysis Phases:**
Bottom-up approach was used for developing applications with software reuse. Design and analysis phase was used for identification of reusable artifacts and other trade-offs related to software reuse. Availability of items for reuse was also considered while designing application with reusable artifacts in it.

**Knowledge Management:**
An important feature of IBM's reuse program was its objective to retain knowledge about reusable artifacts inside the organizations. In very large organizations, people move out or move in. Important and competent members of the team sometimes leave organizations and with them they take important knowledge out of the organization. IBM developed its knowledge management program in order to control this outflow of knowledge. Strategies were developed in order to spread and share knowledge among organization members as well organizational departments. This helped in sustaining the knowledge related to reuse inside the organization. Processes were well documented and people were asked to share there new learn stuff with each other.

**Incentives:**
Incentives were offered to the designers and developers who reuse an artifact or there artifact was reused by someone else. This helped in creating competitive environment related to reuse. Types of incentives were not mentioned in the literature. Idea was to push more and more people in reusing artifacts and developing applications which can be reused.

**Reuse Champions:**
A term often used in literature has its origin at IBM. Reuse champions were introduced; these were the guys who had detail knowledge about the products and how to reuse these products. They were the resource person for different departments in case of any help related to reuse of an artifact.

**Evaluation of Reuse Process:**
Evaluation of reuse processes was performed after a specified time. This evaluation was done keeping in mind the organizational goals and targets. This evaluation helped in improving and generating guidelines for reuse inside organization. Similarly, decisions were made for future reuse related policies.

**Product Lines:**
Concept of product lines was introduced in order to maximize the benefits. Usually products developed by IBM have common features among them. Therefore,
product lines were developed in order to reuse these common features. Continuous efforts were made in order to find the commonality between different products. These common features were made generic enough so that they can be reused in several products.

Motorola

Motorola is an organization which produces applications for telecommunication sector. Few articles have reported cases about Motorola’s success in introducing reuse culture inside the organization. Features related to success or failures of reuse program at Motorola are listed in this section. These factors are identified from articles [33] [25] [4].

Motivation:
Business needs forced Motorola to look for methods which can develop software easily, quickly and of high quality. Therefore, reuse was considered as a major contributor in business needs.

Phased Implementation:
A critical factor in success of reusable artifacts is the phased implementation of reuse strategies. Policies related to reuse were not implemented immediately, they took years to be implemented. Changes were introduced slowly to the organization. This helped to avoid creating a panic situation inside the organization. People accepted the changes since they do not have to change their ways of working completely differently. These phases were spread over number of years.

Bottom Up approach
Bottom up approach meant that organization started introducing reuse culture right at developers level. Changes and guidelines were meant for developers. Training sessions were held for educating developers and designers about software reuse practices. Workshops were conducted in order to share common knowledge about software reuse.

Reuse Repository:
A reuse repository was developed where all the reusable artifacts were stored. Developers and other stakeholders can search this repository and look for artifact for their own need. Guidelines for developing these artifacts were also shared to all the organization staff.

Managements Involvement:
Articles on Motorola’s reuse program reported the involvement of higher management in reuse program. It was considered inevitable and important for any reuse program that higher management takes interest in it. Same was found in Motorola’s case where management was keen to develop a successful reuse struc-
ture inside the organization. Management identified that reuse has a potential to improve organizations finances.

**Pilot Projects for reuse:**
Another idea, practiced by Motorola, was to use small scale projects as a proof of concept before introducing any changes in the organization because of reuse policies. This helped in understanding effect of possible changes and helped in developing measures in order to counter the negative effect. These small projects gave a good idea of future problems related to reuse practices introduced in the organization.

**Reuse of Processes:**
Motorolas research community believed that process could also be reused inside the organization. However, attempts to reuse processes failed because of several reasons listed in articles [25].

**Product Innovation:**
Motorola used reuse as a tool for future innovations. Features of products were introduced in other products in a reusable form which leads to new innovations in these products. These innovations helped in building business value for the old products and in return earning large amount of financial benefits for the organization.

**Reuse of Knowledge:**
Another concept found in literature was reuse of knowledge inside the organization. Knowledge could be in the form of competent architect, developer, and designer or in the form of documents. Efforts were made in order to share the knowledge among the organization.

**Training:**
Designers, architects and other staff members were trained on topics related to software reuse. These training sessions helped in creating awareness among team members about possible problems and standardized solutions of problems.

**Reuse Based Architecture:**
New products were developed based on reusable architecture. Guidelines related to architecture and technology for development were provided by the organization.

**National Aeronautics and Space Administration (NASA)**
NASA has a software development program as an essential to its space mission. Although literature lacks detail about the reuse program present inside NASA, but still few articles describe the general practices present inside the organiza-
Features of reuse program which were found important in these articles are described here. These factors are identified from articles [20] [60] [58].

**Motivation:**
Emphasis of NASAs reusable effort was to find bug-free reusable artifacts which can be reused several times in different products.

**Fault and Change Metrics:**
NASA developed its own fault and change metrics in order to measure the faults found in reused artifacts and impact of these faults. This was necessary for developing error free multi-million dollar products. Metrics measure the impact of change after a fault was fixed and other quality measures. All the measures were specifically developed for reusable artifacts.

**Reuse Working Group:**
A group of experienced members was formed in order to drive reuse goals, set guidelines, monitor the progress, set targets for different units and recommend changes in case of problems. This working group was also responsible for introducing reuse culture inside the organization.

**Software Repository:**
A reuse storage place was developed in order to store reusable artifacts. Retrieving and storing of artifacts in repository has specific guidelines. Based on usage or non-usage of data this repository was updated every year. Old and non-used items were removed while new ones were added.

**Reuse community:**
A community was formed where people can voluntarily share there experiences and ideas on reuse. This prompted some important changes in reuse policy of organization.

**Domain Analysis:**
NASA research community shared an experience where domain analysis was performed in order identify potential reusable artifacts for future.

**Orbotech**
Orbotech [48] is another organization sharing its experience on reusable artifacts. It is an organization which develops CBs, FPDs and imaging solutions for PCB production. Although there was only one article present on its experience but still it is worth mentioning in this report. There experience is important for medium and small size organizations. Important factors from there experience is mentioned here. These factors are identified from articles [48]
New Technology enabling Reuse:
Orbotech molded its process and technology in order to use new technologies which can enable reuse inside organization. New languages and tools were introduced which helped in making reuse easy inside the organization.

Black-Box Reuse:
Producers and consumers of reusable artifacts were asked to use black-box reuse as a guideline. White-box reuse was discouraged. However, code for each black-box reused artifact was provided in order to develop confidence of user and in order to understand the actual working of reused artifact.

Focus on Teams:
Instead of focusing on individuals, for training and other reuse related stuff, Orbotech decided to focus on teams. Each team was responsible for reusing and producing artifact. No single individual was appointed for that matter. This formed a sense of shared responsibility among team members.

3.2.2 Analysis and Discussion
This study pointed out some important results on software reuse in large organizations. These results show trends and important points to consider while introducing reuse process. We can divide these results into three parts: Managerial settings, Industrial setting and reuse related education. This division was made because most of the factors described by organizations fall into these categorize.

Managerial Settings:
Most of the organizations have business driven needs which forced them to implement reuse driven processes inside them. Financial benefits were a major driving force in most of the cases. In NASA’s case, where high value and mission critical software is required; focus was on quality aspect of reuse artifacts. Only after realizing need of reuse, organizations have developed processes required for it. Therefore, management has to be convinced off the benefits of reuse in order to implement reuse related processes.

If management is not willing to invest in software reuse, development process cannot have long term benefits. Reuse required customized settings inside the organization and if organization is not willing to setup resources in order to identify these factors then reuse related benefits cannot be obtained. In a study related to IBM’s reuse processes, non-technical aspects of reuse were considered to be most important.

Management’s role becomes more crucial when a strategy is required for sustaining knowledge about reusable artifacts, which is present inside the organiza-
tion. Management needs to make sure that knowledge sharing and distribution is present and it should be independent of people present inside the organization.

Industrial Settings:

Another result which can be derived from above is the need of proper industrial processes in order to support reuse. Organizations have to implement a mechanism where searching, locating, and obtaining of reusable artifact becomes easy and staff should know where to look for in order to perform these steps. Decision making related to reuse should be done at the start of development. Analysis and design phase can be the most common phase for identification and deciding about any reusable artifact. Documentation was the most common way to describe any reusable artifact. Domain analysis was performed in order to identify artifacts for future reuse.

Industrial settings are technical aspects of reuse program. These settings are important for making reuse possibly, technically and in short span of time. These settings can vary from organization to organization, depending on working methodologies and organizational structure.

Reuse Related Education:
Educating the staff members on the importance of software reuse is equally important. In one of the studies, conducted by author, it was pointed out that designers often do not know about reuse processes present in the organization. Awareness level can be increased by introducing workshops, specific training sessions and other relevant ways in order to emphasis importance of software reuse.

Common Features: Different organizations have various ways to implement reuse processes inside them. However, there are some common features which were identified as common among all the organizations. These features should be considered as guidelines or important steps part of reuse processes. Generically, these processes should be present in all organizations which want to adopt reuse driven development program,

<table>
<thead>
<tr>
<th>Feature</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phased Implementation</td>
<td>All</td>
</tr>
<tr>
<td>Reuse repositories</td>
<td>All</td>
</tr>
<tr>
<td>Management Support</td>
<td>All</td>
</tr>
<tr>
<td>Tailored Reuse Methods</td>
<td>All</td>
</tr>
<tr>
<td>Training</td>
<td>All</td>
</tr>
<tr>
<td>Black-box reuse</td>
<td>All</td>
</tr>
<tr>
<td>Planned Reuse</td>
<td>All</td>
</tr>
</tbody>
</table>

*Table 3.1: Common Factors*
Organization Specific Features:
Importance of tailoring a reuse processes was emphasized and discussed in previous section. Tailoring helps in molding and optimization of processes according to organization needs. Features which were identified as 'organization specific' are listed below:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Innovation</td>
<td>Motorola</td>
</tr>
<tr>
<td>Pilot Projects on Reuse</td>
<td>Motorola</td>
</tr>
<tr>
<td>Fault and change Metrics</td>
<td>NASA</td>
</tr>
<tr>
<td>Focus on Teams</td>
<td>Orbotech</td>
</tr>
<tr>
<td>Reuse of Processes</td>
<td>Motorola</td>
</tr>
<tr>
<td>Reuse community</td>
<td>NASA</td>
</tr>
</tbody>
</table>

Table 3.2: Organization Specific Factors

3.2.3 Recommendations

Increasing demands of software systems are making it hard for the organizations to meet satisfactory level of production [7] [43]. It is hard for the organizations to introduce any rapid changes in order to develop reuse as an integral part of organization. A three phased strategy should be developed for introducing reuse inside organization. First phase should be limited to steps which do not require any organizational changes. These steps can be adopted by staff members without any training and changes. Usually such steps should require less than a year to be implemented. Technical aspects of reuse can be addressed in this phase. Second phase should consist of steps which can be introduced in 1 to 3 years time. However, it is difficult to introduce these basic features at once since it might affect productivity of the organization. Prioritizing these practices should be left on the organizations choice and they can do that based on their immediate need and resources. These practices / guidelines range from technical aspect to managerial aspects. Third phase involves steps which cannot be introduced in short span of time. These are the fundamental changes which should be introduced in small steps and over a lengthy duration. These factors can affect organizations ability to deliver in time if they are introduced rapidly. However, introduction is necessary since it can lead to a systematic software reuse inside the organization. Each organization should consider its own state of affairs before deciding to introduce these factors.

3.2.4 Validity

Focus of report was on industrial experience shared by large organizations. Validity of report is dubious since organizations only share there positive experience
Chapter 3. Background Work

with outside world. Bad experiences are seldom shared. Another important point to consider was the lack of details present in the articles. Articles were not describing their positive features in detail. There focus was more on describing the results and less on how they achieved this result. Factors identified by the author are from limited number of articles. Some factors are identified by author by his own perception of things.

No negative experience or factor was reported by any organizations. There might be several factors which have negative effect on reuse practices. This report therefore misses an important aspect of reuse experience

3.2.5 Conclusion

This research study describes important factors related to reuse practices. These factors were identified from literature review of five organizations. These factors are mostly related to managerial issues and less focused on technical issues. Analysis phase helps in understanding the nature and types of these factors. These factors were of three main types. Common factors found in all organizations were listed. These are applicable for organizations which want to introduce reuse inside their development processes. Recommendations were made for future reuse program to be introduced. These recommendations are based on an iterative process. Validity threats are discussed in respective section.

3.3 Reuse of a black-box component An Industrial Experience

This section has described experience of reusing a software component called Rule Engine in an new product. This new product was developed for an organizations innovation competition and it was based on an innovative idea of merging rule engine of one product with another internal product. This rule engine accepts input and executes rules based on values provided in the input parameters. Furthermore, after getting result from execution of rules some commands are issued to another product called CS. The reused artifact was of foremost importance because whole new product was built upon the results obtained from it. Development team was successful in reusing the rule engine and it worked exactly according to the requirements. New product became success and now the next step was to identify the reasons behind this positive experience. In this report numbers of factors, influential in successful reuse, are described.

Section 2 describes all the success factors which were identified. Section 3 describes suggestions for improving the reuse process. Future work is recommended in section 4. A conclusion of whole proceedings is present at the end.
3.3.1 Factors and Discussion

Researchers have recommended number of practices related to a successful reuse project. Starting point for all these practices is the support of management. If management is not willing to support reuse of software artifact then this reuse process can be ad-hoc and is dependent on individual brilliance of designers and developers [64]. Similarly, Black box reuse is most recommended way of reusing any artifact [55]. A table showing summary of recommended practices from literature and their comparison with practices found in this project is shown below.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Literature Confirmation</th>
<th>Project Confirmation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presence of artifact owner</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Black Box reuse</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Maturity of Artifact</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Management Support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Technological Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Dependency on other modules</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Similarity of Domains</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Willingness of Owner</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Systematic Reuse</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Reduced Search time</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Presence of reuse support team</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

*Table 3.3: Comparison between Literature and Practice*

These factors are closely related to each other. Here are some of the factors which made it possible for the team to reuse Rule Engine (RE) for this prototype project. Each section contains the literature recommendation, experience from the project followed by the discussion.

**Presence and willingness of owner**

According to literature, any reuse component should be owned before it is being reused by anyone [48]. An owner should be assigned to the component and this owner should respond to all the queries related to the component. Owner knows the component inside out and suggests measures in case of any problems. All queries should be directed towards the product owner. A product owner can be a one person or a whole team depending on the complexity of component.

In our case, an owner was available and willing to help. He was available through email all the time and he was quick in replying to the queries. In the initial phase of understanding the component owner was really helpful. He also helped in validating the integration of reuse component with teams prototype. During the intermediate stages he suggested some key points for integrating the
component and these valid points. This factor helped in increasing the confidence level of the team at the initial stages of the development which resulted in developing the product a week earlier than it was expected to be complete.

**Black box and white box reuse**

Literature points out that the software reuse should be in the form of black box [55]. White box reuse is discouraged. Black box reuse not only saves time for the modification but also guarantees a quality product since the reusable component is properly tested before being published.

But in our case the component was between black box and white box reuse. Team was free to customize the component according to the need. Code was provided by the owner and team started working on the prototype by understanding the code and mapping it to the requirements. However, after going through the requirements team felt that they do not need to change any part of the component and the component will be used without any modification. The availability of code made it easy for the team to understand and identify the important parts of code.

Another important point worth mentioning is that the code which was provided to the team was not deployable in application server. It was not previously tested with application server (Glassfish application server [61]) but our requirements demanded a code deployable in glassfish server. By having the component in the form of code, team was able to deploy it and use it according to its own need. If the component was present in the binary form then it would have been very hard for the team to deploy it in glassfish server along with code of newly developed prototype.

**Reduced Searching time**

In literature it is mentioned that searching for a reusable component is often very expensive and takes a lot of effort on the part of team [53]. Therefore, by focusing on only one possible solution team was able to save useful time and resources. Search time was almost none and team members do not have to search and compare any artifact according to the requirements of the project.

Another interesting factor behind the successful reuse is the lack of available alternatives of ERE. Although, open source rule engine might be an alternative for our requirements but author was unable to locate any rule engine which fulfills the requirements of this prototype. Therefore, developers always have in the back of their minds that they have to use the provided rule engine. Team spent their energies on reusing the provided ERE instead of wasting time on searching any other alternative solution.
Chapter 3. Background Work

Working example

A demonstration of integrating ERE with another example code was provided to the team at the start of the prototype development. Although this example was not relevant to prototype but still it removed the fear from teams mind about the component. Secondly, team found the parts of code which needs to be used for integrating the ERE with our prototype. Example was well supported by discussion at the end of the demonstration. Discussion was helpful in understanding the context of the example, relevant parts of ERE and things which should not be changed at all. In literature there was no description present on the effect of working example on reusable component. This area can be investigated in the future research.

Management support

No reuse can be successful without support of management. Nearly all recent studies show the importance of management being aware of possible benefits of reuse. If the management is willing to offer resources and time to the development team chances of a successful reuse increases. [48] [53] [34].

In this prototype, management supported the team by assigning a resource i.e. product owner for helping the team in integrating of ERE. This made a significant difference in this success story as owner of the product has to answer our queries related to product. In another example management assigned the technical experts to help the team when it was stuck in a major problem related to the serialization of custom data types. Technical experts assigned were able to solve the problem and therefore saved a lot of time and effort on part of the team. Similarly, management was keen in knowing the progress of team. Progress was observed on weekly basis, agile approach was followed strictly and management was quick in observing the problems faced by the team. Team got the impression of full support from the management.

Well supported by technology

Object oriented technology was initially hinted as an approach to increase the software reuse. Researches have pointed out this factor in many of the published articles [16] [32]. Java is one of the major languages supporting the object oriented programming approach. Our team used Java for the developing the prototype. The reusable component was also developed in java. The similarity in programming approach made it easy for the developers to integrate ERE with the prototype. Team does not have to bother about the tweaking and modification of original code, the only thing team needed was the understanding of code. After understanding the code, calls to classes were easy to make and objects can be passed from one method to another very easily.
Another important technology factor was the support provided by java to deploy the reused code in application server i.e. glassfish. Any other language might have caused a lot of trouble in deploying the code. The deployment of code was not done in original ERE but it was required by our own prototype. Therefore, java made it very easy for the team to do all this obnoxious stuff.

**Independent module**

Rule Engine (ERE) works as an independent module. It doesn't require services of any other sub-system to perform its main functions. This helps the team in deploying it on a local machine and thus using it without taking care of other sub-systems. Presence of module on a local machine meant that team can experiment on it without worrying about the possible effect over other systems. After deploying the ERE on local machine it worked like a black box where developers do not have to change anything and they were able to fetch the results without any problem. Easy to implement logic for getting the result from ERE, was because of lesser dependency of ERE over other sub-systems.

**Face to Face Meetings**

Often face-to-face meetings are more helpful in identifying the problem as compare to the remote location [9]. In these meetings people can have discussions in a better way as compare to the talking on phone or through Email. In literature related to global software engineering it is recommended to have face to face meetings more often [8]. This helps in improving the quality of the product and it also helps in overcoming the misconceptions among the team members.

ERE was developed in the same location where it was reused. Although it was in different department but sharing the same location helped in finding the owner and helpful resource required for the prototype. Developers and designers can go to the product owner, meet him face to face and discuss everything they wanted to discuss. This helped the team to understand the artifact in a better way. Therefore reuse process was boosted by the presence and availability of face to face meetings with the product owner. Things might have been different if the owner was at some remote location and was available through modern ways of communication. This could have created problems for the team members.

**Similar domains**

The developed prototype and reused component (Rule Engine) were both from the same domain. They both were part of large telecom solution, this helps in reusing ERE. Design of the prototype was made in such a way to accommodate ERE. Architects of prototype knew what is expected of ERE and what value it can provide to the prototype. All this understanding was because of similar in nature products. If ERE was from some other domain, such as embedded
systems, then the reuse would have been difficult. Architects wouldn’t have the knowledge of both domains and therefore possibility of reuse would have been reduced. However in literature we only find attributes supporting the reuse process in one product line [5]. Reusing artifacts from various different products opens up new research questions for the researchers.

3.3.2 Recommendations

Starting point for any reuse artifact is the documentation which is present for it. In this project, the documentation of the reusable artifact was an old version. The document was not relevant to the reuse process. It was providing the detail of functions that reusable artifact can perform. Having a completely relevant and updated version of documentation is a good starting point of any reusable artifact. This will help in reducing the time for understanding part. Similarly, it can address the very basic questions related to the reuse and it can reduce the number of queries directed towards the product owner or product team.

Having a complete team, called as product owner team, is more beneficial and can help the consumer of artifacts in a better way. This team should consist of developer, tester, designer, system analysts and other relevant members. Variety of different members can help in understanding of queries and can answer them in a better way. This team can have a dedicated team lead that is responsible for receiving the queries and directing them to the corresponding team member. The team lead can be dedicated to the artifact on full time bases, but the other team members should only be used when required.

White box reuse should be discouraged and black box reuse should be encouraged. This can reduce the time since understanding of reuse artifact takes more time than any other part of reuse process [53]. For internal reuse of any artifact this can be anticipated in advance and while designing the new release its reuse scenarios should be considered. This can be done with the help of domain experts and experienced designers. Even in the literature we can find methods which support the forward looking reuse process [15].

Neither the consumer of the reuse artifact nor the producer of the artifact found any guidelines for producing and utilizing the artifact for reuse process. Guidelines should be provided which can describe the actual process of reuse. How the producer should take care of the artifact, what needs to be done in order to store that artifact, how a consumer can search for the component and what is the process of obtaining the artifact from producer? These questions need to be addressed by the management and guidelines should be issued for general use.

3.3.3 Future work

This report can become a starting point for achieving the systematic reuse insider the products. Next step should be to fix the problems identified during this small
project and then analyze the factors which are influential for the success of the project. In this way a continuous improvement of the processes can also be achieved. [31].

This project involved the reuse of in-house developed product. In a case where an outside developed reusable artifact is used the factors can vary from the above mentioned factors. In future a detail study can be launched which describes the factors identified from the reuse of third party products (3PP). A comparison report can be an interesting finding for organization and for research community.

Another possible investigation for the future can be the effect of more than one reusable component on any project. A project can be analyzed by reusing more than one component in it. Effect and experience gained from the reuse of each artifact under the same team and same project setting can be very interesting. We can find the variable factor which is common in each of the reusable artifact and the factor which remains the same for each of the artifact.

Working examples of the reusable component not only remove the fears from the mind of developers but also helped in identifying the external interfaces of the reusable component. Further research can be carried out which can identify the time period or number of usages after which an artifact becomes mature for the reuse process.

### 3.3.4 Conclusion

Software reuse promises to provide low cost, high quality and reduced time to market. This report described number of factors which were responsible for a successful reuse in a prototype product. These factors range from technology to management of human resources. Author has described all these factors and is of the view that there is no one single factor which can guarantee success in reuse. Successful reuse will be dependent on many varying factors. However, from the above experience we can list the most important factors which can increase the possibility of success in case of reused software artifact. At the end recommendations are provided which should be introduced inside the organizations in order to make the reuse process more beneficial. These recommendations are related to reuse process as well as reusable product. Lessons learn from this study indicate the need for a more comprehensive study. Here is the summary of the lessons learnt.

Factors which have a positive influence over the project are:

1. Presence of reusable artifact owner
2. Maturity of Artifact
3. Management Support
4. Technological Support
5. Dependency on other modules
6. Similarity of Domains
7. Willingness of Owner
8. Reduced Searching time

Areas where improvement is needed are:

1. White Box reuse should be discouraged
2. Systematic Reuse process should be adopted
3. Guidelines for the reuse process

Following areas can be the focus of future research:

1. Effect of more than one reusable component on any project
2. Factors identified from the reuse of third party products (3PP) and their comparison with currently identified factors.
3. Number of usages or time period after which an artifact becomes mature for reuse.

A section related to future work is also present which describes different scenarios for a similar study. A comparison report of the different scenarios with comparison to this study can also be produced in future.
Chapter 4

Research Methodology

This Chapter describes the research design and how research was conducted during the thesis.

4.1 Research Design

Developing a model, for industrial usage, and evaluating this model involves various research techniques. These research techniques should focus on literature as well as on industrial aspects. Therefore combinations of different techniques were used to answer the research questions. These techniques involve literature review and observations based on action research (since researcher has been actively participating in development projects of organization) and experience of other organizations narrated in literature, section 3.2.

The research design is influenced by the technology transfer method proposed by Tony et.al.[22] and is divided into four phase, as shown in figure. Reason behind this selecting was because of its relevance with industry and simplicity. This design considered real world industry settings and allowed the researcher to be flexible and more realistic when selecting the candidate solution for industry. Other possible alternatives were to carry out this research in more traditional way i.e. industrial case study and then suggesting the findings. However, in such studies the candidate solution is based on theoretical hypothesis and do not consider real world industry settings [56]. Secondly, author has been actively involved in projects inside the organization and believes that various different methods like action research, observations, and literature review will complement each other in providing a reasonable solution.

First phase of the research involved problem understanding and analysis of the problems. This phase took into consideration a number of different factors and used different research techniques to understand the problem context. Second phase consists of development of candidate solution and propose a model for cost avoidance. This solution was based on the preliminary studies and organization settings. Feedback from the practitioners will help in static testing of the framework and will help in tailoring the framework for the current organization. Third phase involved dynamic testing of the framework on various small projects,
which had used various reusable artifacts. Results and analysis achieved from the dynamic testing of project were used to improve the model. Model was released in the fourth phase. This release was complemented by a workshop and a session with actual users of the model.

![Research Design](image)

**Figure 4.1:** Research Design

### 4.2 Conducting the Study

In this section four phases, described in research design, and their sub-phases are described. These phases and sub-phases were influenced by the study described in [22]. Details are as following.
4.2.1 Phase 1: Preliminary investigation

Objective of phase 1 was to analyze the current organization settings and identify key aspects related to the problem. After identifying the problem next step was to formulate a research agenda. All these closely involved the practitioners of industry and regular feedback from the actual users of developed model. Two major steps in this phase are described below:

Step 1: Identify potential improvement areas based on industry needs

For problem identification two steps were performed. In the first step a survey was conducted in the organization on software reuse (section 3.1), in second step several discussion sessions were held with stakeholders inside the organization and third step was to analyze similar experience shared by large organizations in the literature. These sessions were aimed at finding the potential flaws and problem. Second step involves presence of researcher at the site and being part of the development team for different projects. This helped in identifying the affected parts of the organization and in figuring out actual problems related to cost aspects of software reuse.

The reuse survey, present in Section 3.1, was conducted at the designers level. Aim of the survey was to find the awareness levels related to software reuse practices and cost related awareness present at designers level. Results from this survey showed that some extents of software reuse practices are present in the organization. These practices are carried out at individual level. A gap concerning the guidelines for software reuse practices was found.

Other aim of this survey was to find out benefits, in terms of cost, because of software reuse practices. Surprisingly it was pointed out that at the designer level no consideration was given to cost. Mostly designers focused on lesser development time or availability of the artifact as the main reuse design decision. Another gap was found which showed the absence of cost measurement method for any reusable artifact. There was no method which can be used by the managers or designers for calculating the potential costs which they can save or for calculating costs which they saved through any artifact.

Above two survey results were then discussed with upper management and they were not surprised of these results. They were concerned of these two factors and their priority was to have a model which can enable them to measure reuse related costs and also provide guidelines in order to transform opportunistic reuse level to systematic reuse level.

These two problems which were identified: No method present for measuring costs related to reuse, especially costs avoided through software reuse. What steps are required in order to transform an organization from opportunistic reuse to systematic reuse?
**Step 2: Formulate a research agenda**

After discussion with management, research agenda was formulated. An industrial contact was assigned for future discussions. This industrial contact was a useful in pointing out exact needs and requirements. Organizational issues such as getting the required resources were responsibility of this contact. After discussion, with this contact person and practitioners, research objectives were prioritized. Prioritized research agenda was based on organizational needs as well as on the needs of practitioners. After few meetings an informal domain specific vocabulary list was formed between the researchers and practitioners. This list was helpful in communicating the message and information between both groups. Chances of ambiguity were less reduced because of this vocabulary list.

In this case, the primary need identified was to provide a model which can measure the cost avoided through every kind of software reuse artifact. Artifacts can be of four different forms ranging from code to documents, as identified by survey present in section 3.1. Organization does not have any systematic reuse present inside therefore they were mainly concerned with cost avoided through reuse. Less focus was performed on organizational level cost-benefit analysis of reuse.

Secondary need was to provide guidelines which can help in establishing systematic reuse practices. However, it was required that major organizational changes should not be suggested. All the practices should be recommended keeping in mind the organization structure.

**4.2.2 Phase 2: Model Creation**

In this phase actual model was created. This model was then tested in the lab settings as well as presented to the stakeholders. Necessary changes were accommodated in the model, based on the feedback. A simple tool was also developed which implements the model.

**Step 3: Formulate a candidate solution**

Several of meetings with industry contact and observations made by researcher during on site stay led to formation of candidate solution. Some of the characteristic of this solution were as following:

Solution was based on person-hours spent on each task. Reason behind this choice was simple as person-hour was the only metrics collected by the organization. Management thought it would be hard for them to find out other historical data related to each task. Solution should be generic enough to accommodate seven different sites. Each site has its own hourly rate and budget. Expert opinion should be included if no historical data is present of any task. Model developed for measuring cost avoided should not be limited to code reuse. It should measure
four different types of reuse artifacts, identified by initial survey present in section 3.1. Model should be designed keeping in mind the users of the model. In this case users were managers, therefore less technical information should be required to use this model.

Keeping in mind the requirements imposed by the management, a systematic literature review was conducted. Aim of this review was to find out already present models of measuring cost avoided. Chapter 2 contains details of this systematic literature review.

Management asked that guidelines for enhancing software reuse should not involve any major changes. These guidelines should be developed keeping in view the current organization settings and ways of working. Researcher used his working experience inside the organization to formulate the guidelines for organization. These guidelines and their detail are present in section 3.3.

**Step 4: Conduct lab validation**

After creating the model, researcher conducted its lab validation. All the possible values and aspects were considered. Some example scenarios were tried out. It was decided that a tool should be developed in order to use this model. Tool should be simple enough to use and should not involve any complications. A tool was developed which used this model in order to analyze cost avoided because of software reuse. This tool was tested with possible values. Worst case scenarios and beset case scenarios were considered and analyzed. While developing this tool, special consideration was given to the data collection method. Data collection was made easy for globally distributed organizations by collecting data through email.

**Step 5: Perform static validation**

A presentation and demonstration of model and tool was made for the actual users. Results from lab validation were presented. Actual working of tool based model was described to the users. Users objected on the formula used for calculation cost avoidance. They thought that formula used should only be based on number of hours spent on task. Formula should not be based on figures present in literature. Values in formula should only use the data from own organization.

Another feedback was to extend the tool in order to evaluate the results from several different perspectives. Various types of charts, graphs and tables were added in order to analyze the results achieved by software reuse.

Based on this feedback, formula present in the model was fixed. Researcher noted that original formula and previous formula were almost the same but presented in two different ways. However, feedback was entertained accordingly and tool was updated with necessary changes.
Since, industry was less interested in actual implementation of reuse guidelines, therefore no seminar or workshops were held on that topic. However some recommendations were made which are present in section 3.1.

4.2.3 Phase 3: Testing in industrial settings

In this phase dynamic validation of the model was performed in industrial settings. Details of this validation are provided in a section below. This step was helpful in the analysis of the model in real world settings. Limitations and future extensions related to the model were identified and analyzed. Model performed according to the requirements defined initially by the stakeholders and researchers. However there were some recommendations for future extensions but these recommendations were out of scope for this study.

Step 6: Perform dynamic validation

Dynamic validation was performed in three steps. In first step historical data from an old product was used to evaluate the model. In the second step model was tested on a single site with various types of reuse artifacts. Third step involved validation of model at six different sites. This sequence is shown in figure 2.

In the first step, organization stakeholder was asked to provide the data from one project. Data was collected according to the need of model. It was noted that metrics required was easily available and no questions or problems were asked when metric was collected. Metric was total person-hours spent on the product when no old products were reused and other metric was total person-hour spent on product when artifacts were reused from old product. Results from this step indicate that model and tool were mature enough to try in real world settings.

In next step a single site was selected for dynamic validation. This step was performed without any assistance from the researcher. Aim was to see the performance of model and note down the problems faced during the usage of tool. Reuse driver, assigned by the organization, carried out this step. An email was sent to all the stakeholders to identify the artifacts which were reused during last 3 to 4 months. List all these reused artifacts and fill out the data fields for each of these artifacts. These data fields are present in Appendix D. There were seven different artifacts which were identified during one week. These were of 3 different types, environment reuse, framework reuse and reuse of code. No case was reported of documentation reuse. Collection of metrics was easy and results were shared with the management. Later, reuse driver of the organization performed a detailed analysis of the results. This analysis was also performed with the help of developed tool.

In third step, six different sites were contacted to provide data about assets which they have reused recently. These sites were provided worksheets for collecting metrics and guidelines on what to collect and what to consider as reuse.
For the first month two sites replied with data. Fourteen reused artifacts were provided by these two sites. This data was applied on the model and results were analyzed by the practitioners. In the next month 4 different sites reported there reuse cases. There were 24 reuse artifacts reported by these organizations.

Overall management was happy with the working of the model. New aspects of software reuse were discovered while using this model. Organization wanted to measure the costs avoided by the artifacts in terms of number of defects avoided and savings because of the quality of reused artifact. Since this was out of scope of current research and development project therefore it was left to organization for future decisions in this regard. Since management was closely in touch with the researcher, during lab validation and static validation, there were no surprises during dynamic validation.

Figure 4.2: Sequence of Steps in Dynamic Validation
4.2.4 Phase 4: Release of complete solution

After the successful dynamic evaluation of model, it was decided by the industry contact to release this model for the industry. Instead of researchers selling this model to the practitioner, industry contact took control of things. He became relatively expert in using and understanding this model. His experience and discussions with the researchers made life easy for the researchers because he took responsibility of introducing this model to different sites around the globe. Releasing of solution became responsibility of the industry contact.
Chapter 5

Model for calculating Cost Avoidance

After evaluating the pros and cons of nearly all reuse economic models, we reached to the conclusion that a model is required which can help in measuring costs through all kinds of software reuse artifacts. It should not be limited to code reuse. It should provide enough guidelines on how to measure the reuse and how to collect data. Model should only consider cost avoidance and do not take into account cost-benefit analysis or return on investment, which assume that some investment has been done before the gains are measured. Model should not be based on processes such as systematic reuse or ad-hoc reuse processes.

Limitations related to collecting metrics can severely impact the applicability of a model. If a model is too complicated or it is dependent on metrics which are hard to collect then it becomes very difficult for organizations to use the model. These limitations were considered while developing this model.

This section proposes a model which is applicable over all types of software reuse and it can also provide guidelines on what to measure and what to include as software reuse.

5.1 Software reuse metric plan

The first step for this model was to come up with metrics which can be easily collected. Studies have indicated the importance of reuse metrics in order to drive the reuse goals [52]. Therefore it was utmost important for any model to be dependent on simple to collect metrics. The only measure which was common, among most organizations, was person-hours. Most organizations collect person-hours spent on tasks performed. Therefore new model should be based on person-hours as basic metric. Motivation is that by introducing person-hours we can measure costs avoided on every kind of software artifact such as reuse of test cases, development environment frameworks etc. This metrics is easy to collect and no major guidelines are required in order to collect this metrics. A similar approach was used by Amar [1] in their research. Another study by Lim [18] has also used a similar metrics.
5.2 The Model

After deciding about the metric next step was to come up with formula for measuring cost avoided through reuse. Basic inspiration of formula was taken up from Poulins work [53]. However fundamental difference between the two models is that instead of dealing with lines of code, the new formula deals with person-hours spent on reuse artifacts. Benefits of reusing any software artifact can be measured using the formula below:

\[
\text{Cost Avoided} = (\text{Do it yourself cost} - \text{Cost of identification, understanding and integration})
\]

\[
\text{Cost Avoided} = (\text{OI}) \times H
\]

\(O = \text{Person-hours spent on artifact if build from scratch}\)

\(I = \text{Person-hours spent on identification, understanding and integration of reuse artifact.}\)

\(H = \text{Hourly Rate in Currency}\)

Mostly historical data is available if artifact is an internal product of the organization. This data will be used to get person hours spent on artifact when build from scratch. Reuse person hours spent on each artifact includes the time spent on searching, understanding, integrating and testing the artifact [53]. Sum of all these phases will be used as time spent on reusing the artifact.

Characteristics of the model are discussed in sections below:

5.2.1 Third party Reuse products

Internal reuse artifacts are those which are developed inside the organizations. They are not bought form any third party. Person-hours spent on these products are kept in record by organizations in order to pay employees their salaries. Therefore it will be very easy for the organizations come up with the number of hours spent on tasks. Organizations tend to pay their employees based on hours spent on work. Therefore it is much easier and convenient to keep track of time spent on each task. This time spent can be used to measure the costs avoided by that task. Measuring internal reuse savings do not require any new settings before using this model. Organizations usually know about the time and number of resources spent on a product developed inside, they can use this historical data to measure person-hours spent on internal reuse artifacts.

5.2.2 Hourly Rate

Hourly rate can be different in terms of currency and value based on the location of organization. The value of cost avoided, calculated through the formula, will be in the same currency in which hourly rates are mentioned. Model does not
impose any restriction on currency or hourly rates used in the organization. For example, if hourly rate is in Swedish Kronor, e.g. 200 SEK per hour, time spent on artifact when developed from scratch is 600 hours, time spent on reusing the artifact is 100 hours. Then savings are \((600 - 100) \times 200\) = 100 000 SEK. Savings are counted in SEK. Similarly if hourly rate is in US Dollars then savings will be in US Dollars. Same can be true for any other currency.

### 5.2.3 Guidance in data collection

A major factor for introducing the cost model was to provide guidance on how to measure data. As evident from systematic literature review guidelines on what metrics to collect for measuring cost avoided are missing. Organizations are often confused what metrics to collect and what not to collect. Each organization has its own way of understanding a metrics and collecting it. It is necessary for any model to be able to guide on collection of data in order to be effective.

A worksheet is developed which is generic enough to accommodate any organizations without any changes in it. This sheet can be used to collect data. Appendix D shows basic data fields to be collected. Data collecting document is divided into two parts. One part is responsible for basic information of the artifact while other part is responsible for actual metrics collection.

### 5.3 Discussion

This section describes different aspects of proposed framework and provides arguments in support of its use and also discusses potential problems associated which may be present in the model.

#### 5.3.1 Coverage

This model is based on all types of software artifacts. A software artifact can be 'Any piece of software (i.e. models/descriptions) developed and used during software development and maintenance' [62]. A software project can consist of several different artifacts of various types and at various stages. As most of the artifacts can be measured in terms of person-hours spent on them therefore proposed model can be applied on project level as well as on individual products. With the help of this model we can calculate the cost avoided by one artifact as well as cost avoided by several different artifacts of various different products.

#### 5.3.2 Number of reuse instances

Several models take into account number of reuse instances of an artifact [12]. However in this proposed model, counting and using number of reuse instances of
artifact are left on users choice. This change was based on the observation that organizations consider only first time reuse of any artifact as the actual reuse while some organizations consider each reuse instance as a separate reuse case [52].

5.3.3 Cost Time relationship

Time spent on each task (related to reuse) is main factor for increasing or decreasing costs. Organizations pay their employees based on the time they spent working. Model for cost avoidance can use the values based on time spent on each tasks. This model is also based on the same assumption and only considers time spent on each task as the basis of calculating costs avoided. Other models on reuse have used the time based value as the deciding factor in software reuse [12] [2]. Since aim of this model was to come up with simpler and easy solution for cost avoidance, therefore time value for money was not included.

5.3.4 Productivity

Productivity of developers can influence the person-hours spent on tasks. An experienced and competent developer can integrate the component in less time as compare to the new developer. Same assumption can be applied if the artifact is developed from scratch, a young developer might require more time than the experienced developer. The difference, between building from scratch and reusing the artifact, remains the same for both as long as the developer remains the same. Therefore using person-hours as a basic unit for measuring costs is a viable option.

According to Frakes [18] experience level of developer does not affect software reuse. Keeping in mind his claims the productivity factor becomes less significant in software reuse. However, Frakes emphasized on training of developers related to software reuse.

5.3.5 Validity of Model on Different Reuse Artifacts

Bhargav et. al [36] have listed all the software artifacts which can be reused are described in literature. According to their study, fourteen different types of reuse artifacts are present in literature. Table 1 shows possibility of measuring these artifacts in terms of person-hours. Since the basic requirement of the proposed model is to evaluate everything in terms of person-hours, therefore table shows whether an artifact can be measured in person-hours or not.

5.3.6 Generic Model

Because of limited available time model will be tested in one organization only. Since, person-hours are basic metrics collected by most of the organizations there-
Chapter 5. Model for calculating Cost Avoidance

Table 5.1: Applicability of model on different reuse artifacts

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Measured?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td>May be</td>
<td>Amount of time taken to develop the algorithm</td>
</tr>
<tr>
<td>Architecture</td>
<td>Yes</td>
<td>Person-hours spent on developing architecture.</td>
</tr>
<tr>
<td>Data</td>
<td>Yes</td>
<td>Person-hours spent on gathering data.</td>
</tr>
<tr>
<td>Designs</td>
<td>Yes</td>
<td>Person-hours spent on design of product</td>
</tr>
<tr>
<td>Documentation</td>
<td>Yes</td>
<td>Person-hours spent on documentation</td>
</tr>
<tr>
<td>Estimates</td>
<td>May be</td>
<td>Person-hours spent on estimates</td>
</tr>
<tr>
<td>Human Interfaces</td>
<td>Yes</td>
<td>Person-hours spent by the person on an assigned task</td>
</tr>
<tr>
<td>Knowledge</td>
<td>May Be</td>
<td>Depends on type of knowledge i.e. human interfaces</td>
</tr>
<tr>
<td>Models</td>
<td>Yes</td>
<td>Person-hours spent on the development</td>
</tr>
<tr>
<td>Modules</td>
<td>Yes</td>
<td>Person-hours spent on the development of module</td>
</tr>
<tr>
<td>Plans</td>
<td>Yes</td>
<td>Person-hours spent on development of plans</td>
</tr>
<tr>
<td>Requirements</td>
<td>Yes</td>
<td>Person-hours spent on gathering / documenting requirements</td>
</tr>
<tr>
<td>Service Contracts</td>
<td>Yes</td>
<td>Similar to Documentation.</td>
</tr>
<tr>
<td>Test Cases</td>
<td>Yes</td>
<td>Person-hours spent on developing the test cases.</td>
</tr>
</tbody>
</table>

fore model could easily be applied on other organizations. However, this study would be limited to only one organization.

5.3.7 Risks Associated

Organizations have variety of different methods for paying their employees. Person-hours might not be a universal method for keeping track of costs spent on development. This model might not work in that case since Person-hour is the primary metric required for this model. Another possible risk is concerned with complexity of collecting metrics. If a designer is working on more than one task at the same time it would be very hard to keep track of individual task and time and effort spent on that. Third risk is associated with standardization. No standard definition of a reusable artifact can mean different things for different developers. This may lead to confusion.

5.4 Dynamic Validation

The proposed model was developed keeping in mind industry needs. Efforts were made to simplify the model as much as possible. Industry relevant studies become impractical if they are not designed or carried out keeping in mind the problems which industry faces [22]. Therefore, after developing this model first step was to try this model in an industrial environment. In this section details about the dynamic validation are mentioned. This dynamic evaluation was carried out at large telecommunication organization. Following sites were involved in this
evaluation processes, Guangzhou, Shanghai, Gothenburg, Stockholm, Karlskrona and Rijen. Karlskrona site was the main driver for reuse related programs and this research was also driven by Karlskrona site. Basic details of industrial validation are present in Chapter 4.

The dynamic validation was divided into two steps. In first step model was used at a local site. A tool was developed based on the model to assist managers while using this model. During the collection of data researcher was present on site so that in case of any problem managers can contact for help. It also gave a chance to evaluate the model for possible problems and fix these problems before trying the model on several sites in parallel. In second step, tool was used by remote sites for reporting their reuse cases and then cost avoided was calculated. Analysis was carried out on results obtained from the model and recommendations were made for the future use of this mode. Steps followed during validation are described below:

5.4.1 Initiation

A tool was developed which implemented the model. This tool was in the form of Microsoft Excel [10] document with multiple worksheets. Reason for developing an excel sheet, for implementing the model, was to make it easy for managers or reuse drivers to collect data from multiple sites.

Each worksheet was named after each individual site. Two sheets calculate the cost avoidance from different point of views such as cost avoided per site, total cost avoided of all six sites, artifact which avoided the most amount of cost etc. Analysis of cost avoidance was made from various aspects based on the needs of managers. Management can send the excel sheet to different sites and they can fill in the required fields. Management can see the results of individual sites as well as combined result of all the sites.

Four major types for reuse artifacts, used inside the organization, were identified by a survey, present in section 3.1. The data sheets, present in Appendix D, takes care of only four different categorize i.e. Code reuse, Document Reuse (test cases, documentation etc), Reuse of Framework, and Reuse of development environment. This was decided after discussions with stakeholders.

Data Collection

This model requires two basic values in order to be implemented i.e. person-hours spent when artifact is build from scratch and person-hours spent for reusing the artifact. Data collection was to be made from six different sites which were geographically spread across the world. An Email was sent to each of these sites to report their use cases; each site had its own reuse driver for managing software reuse activities. Data of reuse activities was collected for three consecutive months. Each site was asked to identify the reuse cases and report them back to
main reuse driver of the organization. There were in total 24 cases reported from six different sites. These cases were reported through the developed tool.

5.4.2 Data Analysis

Analysis of data was carried out using the developed tool. Tool showed different charts and percentages from different viewpoints. Analysis involved measuring the percentage savings, type of reuse artifact which was extensively reused, time taken to make the artifact reusable and total cost avoided at each individual site as well as combined cost avoided. Main aim of dynamic evaluation was to judge the performance of model and less focus was on the reuse processes inside the organizations. Therefore, lesser emphasis was given to the related to software reuse practices.

5.4.3 Results

Interesting results were found during the evaluation of model. Results showed that, reuse of framework were responsible for most of the savings. Nearly 75% of total savings were because of the reuse of frameworks. However, most number of reuse instances was reported from the category, reuse of environment. Total cost avoided through reuse (till the end of May) was 1.3% of total research budget.

Code reuse took most amount of time for making the artifact reusable (searching, understanding and integration). Reuse of environment took least amount of time. No case was reported for the reuse of documentation. A possible reason for it could be the limited amount of time (2 months) provided to the sites in order to identify and collect cases. Secondly, as evident from literature review, practitioners and researchers sometimes only focus on executable artifacts or artifacts which directly relate to software development. However, further investigation is required in order to analyze this lack of cases reported in documentation reuse.

Few interesting facts were found during the analysis phase. Different articles [16] [3] claimed that it takes around 20% effort of original artifact to reuse a black box code artifact. In our case white box artifacts, on average, took around 33% effort of original artifact in order to reuse it. However, there were artifacts which took as much effort to reuse them as it was required to build them from scratch. It was interesting to find that, size of artifact has no linear relation with the savings. It was noted that medium size artifacts require more effort to reuse them as compare to large or small artifacts. This pattern was even more evident in cases where development environment was reused. Total savings were about 1.3% of total development budget. These were close to the figure provided in the literature [1].

Some important findings are listed here:

Reuse of Code took most amount of time as compare to the reuse of other artifacts. Reuse of medium size artifacts (150–300 person-hours) took most amount
Chapter 5. Model for calculating Cost Avoidance

of effort in order to reuse these. Reuse of frameworks saved most amount of money. White Box reuse takes around 33% of effort, however some code components takes as much time as building from scratch. Development environments were the most reused artifacts.

These findings were gathered from 24 different reuse cases collected in the organization. These cases were carried out by different subjects with various different backgrounds. These finding are not limited to a particular department or person, these are generalized results from various reused products. However, aim of this study was not to investigate reasons behind these findings, this study is limited to cost related aspects of software reuse.

5.4.4 Evaluation of Model

Feedback from different stakeholders was collected in a meeting. Managers from remote sites emailed their feedback about the usage and results received from the model to the reuse driver of the organization. A one hour session was conducted in which three managers gave their feedback about the working of proposed model. Overall feedback was positive and model was able to achieve what was required. Management was happy with the performance and results obtained, however important suggestions were made about measurement of maintenance costs.

Simplicity and ease of data collection was one of the major aims of model. Basic aim was to provide assistance in collection of metrics so that resources were not wasted on collecting of wrong metrics or useless metrics. Managers found it easy to fill out the data collection sheet, not many questions were asked on this issue. Gathered data was easily combined for analysis and presented to higher management. Usage of email for sending data collection sheet avoided any confusion. Person responsible for filling the data sheet with required reuse cases were not troubled by definition any field. Definition and example of each column was provided in the data sheet, this helped in reducing the confusion regarding common understanding of terms.

Second aim of the proposed model was, to measure each and every kind of reuse artifact. Four different types of artifacts were measured during this dynamic evaluation. These artifacts vary from code to documents. In previous research work no account of environment reuse was found. This model was able to measure costs avoided through reuse of such artifacts. The four artifacts measured were of different types ranging from code to documents but model performed at satisfactory level for each one of these.

Results obtained showed that in first 5 months costs avoided through reuse were around 1.5% of total research budget. These numbers were compared with the claims present in literature [1] and this 1.5% value was found to be a realistic number. According to literature in organizations with ad-hoc reuse processes have savings of around 5% of development budget per year. A separate study was conducted on the organizations reuse process and it was found to be ad-hoc
reuse process. Hence results obtained are justifiable keeping in mind the previous study.

While model was used in industry for 2 months. Suggestions were made regarding the expansion of this model. Some stakeholders wanted it to measure cost avoided through maintenance. According to previous studies costs avoided through maintenance are the major costs avoided through software reuse [17] [18]. These maintenance costs can only be calculated if the reused software is used for lengthy duration of time. Only then we can have number of hours which were avoided in terms of maintenance costs. However, aim of the model was to measure the costs avoided through reuse artifacts. It was not planned to accommodate the maintenance costs and model do not accommodate such costs. Future work discusses further research on this topic.

**Comparison with previous models on Cost Avoidance**

Two previous models [53], [12] were based on lines of code. These two models cannot measure the artifacts other than code. Therefore, our model is better in terms of coverage of various reusable artifacts. In literature, we do not find any experience of dynamic validation provided by these models, as stated in chapter 2. Hence, improvement is in the area of coverage and generic nature of this model.

Third model [1] related to cost avoidance was very similar to this model. However, our model is simple and easy to apply in terms of metrics it requires. Old model [6] required various metrics which were difficult to collect. Hence, simplicity of use is major improvement in the old work. Other features remain very similar to this model.

**Comparison with COCOMO**

COCOMO Intermediate involves lines of code as its basic estimation formula. However, it does not consider documents, test cases etc. Cocomo basic was an old model which did not take into account the new development technology and provides its estimation based on three types of software artifacts only [63]. These estimates may be valid for code reuse but various other types of artifacts cannot be measured by these mdoels. In this study we aim to measure costs which were avoided previously, however COCOMO measures effort which will be required in future.
Chapter 6
Guidelines for introducing reuse culture

Increasing demands of software systems are making it hard for the organizations to meet satisfactory level of production [7] [43]. It is hard for the organizations to introduce any rapid changes in order to increase profits. Organizations present at the opportunistic level of software reuse usually have no planned reuse in their products. Reuse present in such organizations is because of individual efforts. Although management of organizations wants to increase reuse but often they do not have enough resources and time for that. Therefore simpler guidelines are required which can guide organizations to more systematic form of software reuse. In this section these guidelines are discussed.

As evident from the basic formula used in the model for increasing the savings through software reuse, costs of reusing an artifact should be reduced. This will increase the savings of organization and could reduce the lead time as well. Four phases required for reusing software were described by Poulin [53]; these phases are searching, locating, understanding and integration (adaptation) of reuse artifact. Lesser the time spent on searching, locating, adapting and testing more will be the profit. Therefore necessary steps are required in order to reduce costs related to these four factors. After going through the experience shared by several studies on success and failure factors of software reuse [64] [55] [21] [29] [13] [23], a three phase strategy was developed. Author of this research shared his own experience in a study about an organizations reuse success factors in section 3.3 as well. This research also helped in understanding the problems faced by very large organizations related to reuse. The three phases are described as following:

6.1 Initial Strategy

Initial strategy involves steps which should be taken immediately. These steps are not difficult and require managements commitment to the reuse efforts. An organization can introduce these steps in a year long time. No major time consuming activity is present in this phase. These steps are described as following:
6.1.1 Small scale domain analysis

At this initial level organizations have low level of reuse. Therefore first step should be to identify the reusable components present inside the organization. Identification of reusable component can be tricky. Matching the requirements is a good way to find similarity between the artifacts. A similar approach was discussed by Martinez et al. [44]. A resource at a designers level should be assigned for that purpose. That resource should be expert in technical aspects of products. Initially only small size artifacts (2000 SLOC- 3500 SLOC) should be identified. Small artifacts are easy to adapt, according to the result of this very study in analysis section.

6.1.2 Introduction of component library

A component library should be introduced which can store artifacts as well as meta-data about the artifact. Component library should store the requirements, source code and information of the contact person in the library [13]. Considerable amount of publicity should be done about this library. During a recent survey, present in section 3.1, it was noted that developers often do not know about any library while managers know about it. Since it is the developers how perform the actual work therefore awareness level of developers should be increased about the library and its usage.

6.1.3 Contact person for reusable artifact

Adapting and understanding an artifact is a difficult task. Help in this regard reduces the time to adapt the artifact. In the initial stage, when no systematic processes for software reuse are present, a contact person should be assigned. This contact person should preferably be the developer of that artifact. He should be able to take some time out from weekly routine and answer questions related to that artifact. Each artifact should have its own contact person. For seeking help related to that artifact, the contact person should be contacted through email.

6.2 Short-term Strategy (1 to 3 years time)

Short term strategy involves practices which can be introduced in 1 to 3 years time. These are generally agreed upon features [68] [20] [33] [48] listed by studies. However, it is difficult to introduce these basic features at once since it might affect productivity of the organization. Prioritizing these practices should be left on the organizations choice and they can do that based on their immediate need and resources. These practices / guidelines range from technical aspect to managerial aspects.
6.2.1 **Encourage Object Oriented based design**

Object oriented design makes reuse relatively easy as compare to the procedural programming style. Objects and classes are easily reused if they have less dependency over other classes or objects. Several research studies supported the idea of objected oriented programming for software reuse [37] [11]. Organizations which are using Java, C# or C++ as their development language can easily adapt to pure object oriented designs. Lesser time for adaptation of artifact means more profit for the organization. It also reduces the complexity level of reusable artifact.

6.2.2 **Optimize component library**

Different organizations have different ways of storing and documenting an artifact. Based on the needs and practices of the organization, these library artifacts should be stored. Some organizations focus on good search tool for fetching meta-data about the artifact while other organizations may be interested in actual requirements of the artifact. Therefore, based on the work habits and needs, library should be modified. Before this modification extensive feedback should be collected from all the stakeholders. This feedback should span over a year. Analysis of feedback should be done and steps for the improvement of library should be finalized. This will help in shortened search time which is a big factor in reuse related costs [13].

6.2.3 **Store medium size components**

In the initial phase, small phase artifacts were recommended. However, for more profitability medium and large size artifacts should be used. After learning from the experience of storing small size artifact, organizations should analyze this experience and try to come up with improved way of storing artifacts which are larger in size. Larger artifacts are difficult to adapt and complexity level increases with the size of artifact. Therefore medium size artifacts should be adapted for reuse before storing in the library. More focus should be paid on storing medium size component as compare to larger size components.

6.2.4 **Encourage black-box reuse**

Adaptation of artifact for the reuse purpose is a complex job. Understanding the code written by someone else is never an easy task. Therefore black box reuse should be encouraged inside the organization. Any artifact which is stored or retrieved from the library should be used as it is. This will reduce the adaptation costs and it will also increase quality aspect of the artifact [48].
6.2.5 Design applications keeping in mind their reuse

At the early stages of development it is difficult to predict the future reuse of artifact. However, with the passage of time guidelines for developers should be introduced in order to develop reusable components. These guidelines should be purely technical and should correspond to the domain and programming language used for development. This will help in producing artifacts which are already adapted for reuse, reducing time for adaptation [20] [48].

6.2.6 Documentation, groups, forums on reuse

One of the objectives of short term strategy is to become independent of human resource. Human resource often move in and out of the organization, taking with them useful knowledge of products. Documentation is considered as a useful way of storing information. Documentation related to software reuse should be started in this phase. Document should tackle the most common problems faced while reusing the artifact.

Forums and groups are modern ways of sharing knowledge. They have been found as a good and quick way of seeking help. Developers community realized the importance of forums years ago. Organization can introduce reuse groups and forums where people can discuss problems and find a solution. Similar experiences are shared by other large organizations as well [20].

6.2.7 Incentive on Reuse

In order to increase the reuse level and reuse awareness, incentives should be introduced in the organizations. Incentives should be given to the developers who have reused more artifacts or who has introduced more artifacts in the library which were reused by several developers. This can lead to a healthy competition inside the organization.

6.3 Long-term Strategy (3 to 5+ years time)

Long term strategy involves steps which cannot be introduced in short span of time. These are the fundamental changes which should be introduced in small steps and over a lengthy duration. These factors can affect organizations ability to deliver in time if they are introduced rapidly. However, introduction is necessary since it can lead to a systematic software reuse inside the organization. Each organization should consider its own state of affairs before deciding to introduce these factors.
6.3.1 Development language and framework

Languages which supports object oriented design and component based artifacts should be encouraged. Reuse of code is easy if the language and design has less coupling and more cohesion inside an artifact. This helps in producing black-box artifacts which are easy to reuse. Since introduction of new language in the organization requires new experience and skills of developers therefore this change cannot be rapid. Long term planning and execution is required in order to change development language of organization. Exporting components from one platform to another platform has been made easy through recent developments in frameworks. Now components developed in one programming language can be reused in another programming language without too much adaptation costs [59].

New and improved frameworks like Java Enterprise, .Net Framework [41] [59] allow developers to develop deployable objects. These objects can work independently and do not require adaptation for reuse purposes. However introduction of such frameworks require expertise at developer and designer levels. This cannot be achieved over one or two years. Organization has to invest in order to improve the competence level present in the organization.

6.3.2 Refactoring Contents of Component Library

Component libraries should be regularly updated. Old artifacts which are not reused in the past should be removed. New components should be regularly introduced [13].

6.3.3 Product Line Engineering

Very large organizations with similar product can introduce product line engineering in their processes. This considerably improves profit levels and reduces time to market a product. Introduction of product line is a lengthy and hectic process. Ways of introducing product line engineering is itself a wide research area. More information on that can be found in literature. [66] [38]
Software reuse was claimed to be silver bullet [20] required by software industry. Despite of several years of research, management of organizations still need to be convinced about the importance of software reuse and its financial benefits. Organizations with ad-hoc reuse processes find it difficult to measure potential gains they had from the reuse of software artifacts. Varieties of cost models for software reuse are found in literate. These models try to provide solution of measuring reuse benefits. However these models are either based on assumptions, such as presence of systematic reuse processes and investment on reuse, or these models are too complex to apply in organizations, which collect limited amount of metrics. The model proposed here tries to overcome these difficulties by providing a model which is simpler in understanding and also provides guidelines which type of metrics to collect and how to collect the metrics. Secondly the model covers 13 different types of reuse artifacts other than code reuse.

The proposed model has limited applicability as its focus was on cost avoidance through software reuse. It does not cover the return on investment or cost-benefits of software reuse. However, this model can measure savings from all kinds of commonly reused artifacts. Two metrics are required in order to apply this model. First metric should be the time spent on artifact when it was build from scratch and other metric was the time spent on artifact in order to reuse it. This reuse time includes time spent on identification, understanding and integration of artifact.

Dynamic evaluation of model was done in industry. A tool was developed which implements this model and also used for data collection. An organization was selected to run the first evaluation of the model. Six different sites were contacted to report their reuse cases using the implemented tool and guidelines provided to them. There were in total 24 cases reported from six different sites. Savings reported were around 1.5.

Dynamic evaluation pointed out that model should accommodate the costs avoided through maintenance. However this requires lengthy time period to collect further data. Scope of this study was limited to the direct savings through reused artifacts; therefore in future this model needs to accommodate maintenance savings.
Increasing the profits is the main objective of any successful organization. Guidelines for increasing the profits through software reuse are provided in this article as well. These guidelines are generally agreed upon success factors, mentioned in the literature and based on a study conducted by the author himself.
## Appendix A

### Results after Initial search and search strings

<table>
<thead>
<tr>
<th>Search Term</th>
<th>Total Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Reuse</td>
<td>2755</td>
</tr>
<tr>
<td>Software Reuse Cost Model</td>
<td>453</td>
</tr>
<tr>
<td>Reuse Cost Model</td>
<td>18</td>
</tr>
<tr>
<td>Software Reuse Cost measurements</td>
<td>403</td>
</tr>
<tr>
<td>Software Reuse Cost measuring framework</td>
<td>0</td>
</tr>
<tr>
<td>Software Reuse Cost Estimation</td>
<td>232</td>
</tr>
<tr>
<td>Software Reuse Savings</td>
<td>227</td>
</tr>
<tr>
<td>Measuring software reuse benefits</td>
<td>233</td>
</tr>
<tr>
<td>Software Reuse Economics</td>
<td>281</td>
</tr>
<tr>
<td>Reuse Economics</td>
<td>132</td>
</tr>
<tr>
<td>Software Reuse Benefits</td>
<td>243</td>
</tr>
<tr>
<td>Reuse Benefits</td>
<td>69</td>
</tr>
<tr>
<td>Software Reuse Cost Avoidance</td>
<td>227</td>
</tr>
<tr>
<td>Reuse Cost Avoidance</td>
<td>1</td>
</tr>
<tr>
<td>Reuse return on investment</td>
<td>1053</td>
</tr>
<tr>
<td>Software reuse return on investment</td>
<td>243</td>
</tr>
<tr>
<td>Reuse Finance</td>
<td>0</td>
</tr>
<tr>
<td>Reuse Cost Estimation</td>
<td>6</td>
</tr>
<tr>
<td>Reuse Cost Assessment</td>
<td>8</td>
</tr>
<tr>
<td>Software Reuse Cost Estimation</td>
<td>232</td>
</tr>
<tr>
<td>Software Reuse Cost Assessment</td>
<td>336</td>
</tr>
<tr>
<td>Software Reuse Business</td>
<td>257</td>
</tr>
<tr>
<td>Reusability cost</td>
<td>50</td>
</tr>
<tr>
<td>Software Reuse cost measurement method</td>
<td>0</td>
</tr>
<tr>
<td>Software Reuse cost measurement technique</td>
<td>0</td>
</tr>
<tr>
<td>Software reuse cost reduction</td>
<td>227</td>
</tr>
<tr>
<td>Software reuse price</td>
<td>227</td>
</tr>
<tr>
<td>Tracking Reuse Cost</td>
<td>3</td>
</tr>
<tr>
<td>Monitoring Reuse cost</td>
<td>1</td>
</tr>
<tr>
<td>Reuse Cost-benefit</td>
<td>17</td>
</tr>
<tr>
<td>Software Reuse ROI</td>
<td>230</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Software Reuse Financial aspect</td>
<td>436</td>
</tr>
<tr>
<td>1 AND 10</td>
<td>267</td>
</tr>
<tr>
<td>1 AND 3</td>
<td>13</td>
</tr>
<tr>
<td>1 AND 13</td>
<td>227</td>
</tr>
<tr>
<td>1 AND 12</td>
<td>14</td>
</tr>
<tr>
<td>16 AND 1</td>
<td>227</td>
</tr>
<tr>
<td>Total</td>
<td>9348</td>
</tr>
</tbody>
</table>
## Appendix B

### Software Reuse Cost Models

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Measuring the Roi of Reuse, Object Magazine 4, 3, 4854.</td>
</tr>
<tr>
<td>8</td>
<td>Return on Investment of Reusable Components: Analytical and Empirical Approaches, Technical Report, University of Ottawa, Ottawa, ON, Canada.</td>
</tr>
<tr>
<td>11</td>
<td>Value Based Software Reuse Investment, Annals of Software Engineering 5, 552.</td>
</tr>
<tr>
<td>12</td>
<td>Technical Report, Center for Software Engineering, University of Southern California, Los Angeles, CA</td>
</tr>
<tr>
<td>13</td>
<td>Software Reuse Key to Enhanced Productivity: Some Productivity Models, Information and Software Technology 31, 5, 258267.</td>
</tr>
<tr>
<td>14</td>
<td>Making Reuse Cost Effective, IEEE Software 8, 1, 1324.</td>
</tr>
<tr>
<td>15</td>
<td>Economics of Reuse: Issues and Alternatives, Information and Software Technology 32, 10, 643652.</td>
</tr>
<tr>
<td>16</td>
<td>Managing Software Reuse Economics: An Integrated ROI-based Model</td>
</tr>
<tr>
<td></td>
<td>Appendix B.  Appendix B</td>
</tr>
<tr>
<td>---</td>
<td>-------------------------</td>
</tr>
<tr>
<td>18</td>
<td>Evaluating software reuse alternatives A model and its application to an industrial case study</td>
</tr>
<tr>
<td>21</td>
<td>&quot;Measuring the benefits of software reuse - Examining three different approaches to software reuse.&quot; Dr Dobbs Journal 30(6): 73-76.</td>
</tr>
<tr>
<td>26</td>
<td>Aligning the economic modeling of software reuse with reuse practices, Information and Software Technology, Volume 50 , Issue 7-8 Pages: 753- 762 ,2008 ISSN:0950-5849</td>
</tr>
</tbody>
</table>
Appendix C

Articles without full text available

<table>
<thead>
<tr>
<th>No</th>
<th>Article</th>
</tr>
</thead>
</table>


Appendix D

Description of each field is given below:

Case No: Listed for keeping track of numbering and for reference in other reports.

Name of the reuse artifact: The name used for artifact in the organization.

Category: Artifacts can be categorized based on the requirements of organization. Each organization has its own products and these products vary in different domains. For example in telecommunication systems frameworks are often used extensively but organizations dealing with low level system architecture are more concerned with reuse of algorithms. Therefore this category is customizable according to the requirements of organization.

Origin: Original product from which artifact was taken. It can be name or version which is used inside the organization.

Reused In: The final product in which software artifact was reused. It can be the name or version which is used inside the organization.

Quarter: Depends on organizations policy for generating reports. Value of this column can be replaced by month or year.

Calculations: Man-hours spent on artifact when build from scratch. In case where no historical record of man-hours is present we can use expert opinion to estimate the number of hours. Man-hours spent on artifact when reused (identification, understanding, integration and testing of artifact).

Comment: For providing any useful comments. Used if some clarification needs to be addressed.
Appendix E

Definitions

**Reuse:** To use again, especially after salvaging or special treatment or processing.

**Guidelines:** A detailed plan or explanation to guide you in setting standards or determining a course of action.

**Profitability:** ”The quality of affording gain or benefit or profit”.

**Person-Hour:** A person-hour is the amount of work performed by an average worker in one hour.

**Profit:** ”An excess of the receipts over the spending of a business during any period”.

**Cost Avoidance:** Preventing any costs from happening or occurring.
References


