Evaluating Evolutionary Prototyping for Customizable Generic Products in Industry (TAT AB)

Vickey Kamlesh and Shoaib Ahmad
This thesis is submitted to the School of Engineering 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 for two students.

**Contact Information:**
Author(s):
Vickey Kamlesh Wadhwani
E-mail: vickeywadhwani@gmail.com

Shoaib Ahmad
E-mail: shoaib.ahmed83@gmail.com

External advisor(s):
Karl Anders
TAT AB
Address: Norra Vallgatan 64, 211 22 Malmö, Sweden
Phone: +46 40 109 700

University advisor(s):
Dr. Mia Persson
Department of Systems and Software Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
ABSTRACT

Software products can be categorized into three types namely bespoke, market driven and customizable generic products. Each of these products is facing different problems in their development and to address these problems different software process models have been introduced. The use and validation of different software process models for bespoke and market driven products have been discussed in earlier work. On the other hand, less attention was paid to the customizable generic products. Our thesis will fill this gap by conducting a case study on evolutionary prototyping (EP) for customizable generic products. The main aim of the thesis is to make an initial validation of EP for customizable generic products. In order to fulfill the aforementioned aim we performed a literature study on prototyping and EP, together with development of two customizable generic products. During this development process, we used approach of EP. The results from our investigation will provide researchers and practitioners with a deep insight to the EP and also to guide them in making decision regarding the use of EP. The main findings from our investigation are as follows:

- EP is not used standalone as a software process model. Rather it is used as a concept that can be augmented with some iterative software process model.

- Negative and positive aspects of EP were highlighted by discussing situations where it could be a better choice, with its advantages and disadvantages.

- An initial validation was performed on EP for customizable generic products. Reported results from this case study show that the selected approach is a good choice when you want to have innovative product, clear ambiguous and sketchy requirements, discover new requirements, save resources of software testing, involve and satisfy customer. EP shows vulnerabilities in documentation of product and quality of code.

Keywords: prototyping, evolutionary prototyping, customizable generic products, scrum
ACKNOWLEDGEMENTS

We would like to thank everyone that has contributed to our master thesis, especially:

- Our advisors Dr. Mia Persson at Blekinge Institute of Technology (BTH) and Karl Andres at TAT AB (Tenk), for the support, guidance and constructive criticism.

- We are thankful to Hampus Jakobsson and Josef Granqvist for their support and active participation during our work at Tenk.

- We are also very thankful to all members of Tenk for their support and help in the completion of our master thesis.

- Last but not the least we are thankful to our parents and entire extended family for supporting us in our study.
# CONTENTS

1. Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1. Gap and its Importance</td>
<td>1</td>
</tr>
<tr>
<td>1.2. Aims and Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3. Research Questions</td>
<td>2</td>
</tr>
<tr>
<td>1.4. Expected Outcomes</td>
<td>2</td>
</tr>
<tr>
<td>1.5. Research Methodology</td>
<td>2</td>
</tr>
<tr>
<td>1.6. Research Setting</td>
<td>3</td>
</tr>
<tr>
<td>1.7. Methodology for Results</td>
<td>3</td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
</tr>
<tr>
<td>1.8. Structure of Thesis</td>
<td>3</td>
</tr>
</tbody>
</table>

2. Background

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1. History of Software Development</td>
<td>4</td>
</tr>
<tr>
<td>2.2. Software Process Models</td>
<td>4</td>
</tr>
<tr>
<td>2.2.1. Prescriptive Models</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1.1. Waterfall Model</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1.2. Incremental process Models</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1.3. Evolutionary Process Models</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1.4. Specialized Process Models</td>
<td>5</td>
</tr>
<tr>
<td>2.2.1.5. Unified Process</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2. Agile Models</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.1. Extreme Programming (XP)</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.2. Adaptive Software Development (ASD)</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.3. Dynamic System Development Method (DSDM)</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.4. Scrum</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.5. Crystal</td>
<td>6</td>
</tr>
<tr>
<td>2.2.2.6. Feature Driven Development (FDD)</td>
<td>6</td>
</tr>
<tr>
<td>2.3. EP</td>
<td>7</td>
</tr>
<tr>
<td>2.4. Customizable Generic Products</td>
<td>7</td>
</tr>
</tbody>
</table>

3. Overview of Prototyping and EP

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1. Methods, Tool and Approach for Prototyping</td>
<td>9</td>
</tr>
</tbody>
</table>
3.1.2. How Prototyping Was Applied in Software Development Model
3.1.3. Analysis of Prototyping in Industry
3.1.3.1. Evaluating prototyping by different attributes
3.1.3.2. Factors Given no importance in the Evaluation of Prototyping
3.1.3.3. Evaluating Effect of Prototyping on Software Quality
3.1.4. Pitfalls of EP
3.1.5. Future Benefits for Company Using Prototyping
3.1.5.1. Awareness
3.1.5.2. Erasing Myths
3.1.5.3. Technical Enhancement
3.1.5.4. Personnel and Organizational Issues
3.1.5.5. Complementary Process and Technologies
3.2. Prototyping from the Period of 1996-1999
3.2.1. Approach
3.2.2. Comparing EP with Waterfall and Agile
3.2.3. Benefits of EP
3.2.4. Drawbacks
3.2.5. Practices in Industry
3.2.6. Prototyping User Interfaces
3.2.7. Evolution till 1999
3.3. Prototyping from the Period of 2000-2004
3.3.1. Why process Differs in Software Development
3.3.2. Agile Methodology and Prototyping
3.3.3. Prototyping and Requirements Engineering
3.3.4. Prototyping and Software Testing
3.3.5. Prototyping and Web Applications
3.3.6. Evolution till 2004
3.4. Prototyping from the Period of 2005-Onward
3.4.1. Prototyping in SDLC
3.4.2. Early acceptance Testing of Mobile Applications Using Rapid Prototyping Framework

3.4.3. Prototyping for Web Applications and Computer Applications

3.4.4. Use of Prototyping for User Interface Acceptance Testing

3.4.5. Evolution till Now

3.5. Conclusions

4. Alternatives of EP for Software Development

4.1. Alternatives of Evolutionary Process Model for Software Development and their Comparison

4.1.1. Prescriptive Models

4.1.2. Agile Models

4.2. EP and Agile

4.3. Rationale for the Selection of EP

4.4. Conclusions

5. Scope of EP

5.1. Situations where EP is Suitable

5.1.1. Ambiguity and Change in Requirements

5.1.2. Short Time to Market

5.1.3. Managing Risk of Innovation

5.1.4. Too Many Stakeholders

5.1.5. Software Testing at Early Stages of Software Development

5.1.6. Testing User Interfaces
5.2. Advantages and Limitations of EP
5.2.1. Advantages of EP
5.2.1.1. Removing Misunderstandings
5.2.1.2. Customer Satisfaction
5.2.1.3. Problem Identification at Early Stages
5.2.1.4. Easy Adaptable
5.2.1.5. Better Product Validation
5.2.1.6. Flexibility in Phases
5.2.1.7. Focus on Working Piece of Code and Design Flexibility
5.2.2. Limitations of EP
5.2.2.1. Poor Quality Standards of Code and Maintainability
5.2.2.2. Requires Experienced and Skilled Manpower
5.2.2.3. Bad Estimation of Cost and Time
5.2.2.4. Time Consuming
5.2.2.5. Too Much Customer Involvement
5.3. Customizable generic products and EP
5.4. Conclusion

6. EP and TAT AB
6.1. TAT Introduction
6.1.1. TAT Products
6.1.1.1. TAT Cascades™ UI Solutions
6.1.1.2. TAT Motion Lab™
6.1.1.3. TAT Cascades™
6.1.1.4. TAT Kastor™
6.1.2. Processes at TAT AB
6.2. Tenk AB
6.2.1. Processes at Tenk AB
6.3. Applicability of EP at Tenk
6.3.1. EP and Our Projects at Tenk
6.4. Conclusions

7. Initial Validation of EP for Customizable Generic Products
7.1. Social Playlist and Social Location as Customizable generic products
7.2. Research Setting (Conditions,
Environment, and Data of EP:

7.2.1. Conditions and Environment of Development: 46

7.2.1.1. Conditions and Environment of Social Playlist: 46

7.2.1.2. Conditions and Environment of Social Location: 47

7.2.2. Data Collected From Social Playlist and Social Location Projects: 49

7.3. Evaluating the Results of EP:

7.3.1. Time Estimation: 50

7.3.1.1. Time Estimation for Social Playlist and Social Location Projects: 50

7.3.2. Documentation:

7.3.2.1. Documentation of Social Playlist and Social Location Projects: 51

7.3.3. Software Testing:

7.3.3.1. Testing in Social Playlist and Social Location Projects: 51

7.3.4. Quality of Code:

7.3.4.1. Quality of Code in Social Playlist and Social Location Projects: 51

7.3.5. Quality of Product:

7.3.5.1. Quality of Product in Social Playlist and Social Location Projects: 52

7.3.6. Customer Involvement and Satisfaction:

7.3.6.1. Customer Involvement and Satisfaction in Social Playlist and Social Location Projects: 52

7.3.7. Team Effort:

7.3.7.1. Team Effort for Social Playlist and Social Location Projects: 53

7.3.8. Delivery Time:

7.3.8.1. Delivery Time of Social Playlist and Social Location Projects: 53

7.3.9. Change management and Overload: 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.9.1. Change Management and Overload in Social Playlist and Social Location Projects</td>
<td>53</td>
</tr>
<tr>
<td>7.3.10. Effort Distribution for Social Playlist and Social Location Projects</td>
<td>54</td>
</tr>
<tr>
<td>7.4. Results Analysis</td>
<td>54</td>
</tr>
<tr>
<td>7.4.1. Innovative Ideas</td>
<td>54</td>
</tr>
<tr>
<td>7.4.2. Ambiguous and Sketchy Requirements</td>
<td>55</td>
</tr>
<tr>
<td>7.4.3. Discovery of New Requirements</td>
<td>55</td>
</tr>
<tr>
<td>7.4.4. Customer Involvement and Satisfaction</td>
<td>55</td>
</tr>
<tr>
<td>7.4.5. Software Testing</td>
<td>55</td>
</tr>
<tr>
<td>7.4.6. Software Quality</td>
<td>56</td>
</tr>
<tr>
<td>7.5. Threats to Study</td>
<td>56</td>
</tr>
<tr>
<td>7.5.1. General Validity Threat</td>
<td>56</td>
</tr>
<tr>
<td>7.5.2. Threat in Findings</td>
<td>56</td>
</tr>
<tr>
<td>7.6. Conclusions</td>
<td>56</td>
</tr>
<tr>
<td>8. Conclusions and Future Work</td>
<td>57</td>
</tr>
<tr>
<td>8.1. Answers to Research Questions</td>
<td>58</td>
</tr>
<tr>
<td>8.2. Conclusions</td>
<td>58</td>
</tr>
<tr>
<td>8.3. Our Findings</td>
<td>58</td>
</tr>
<tr>
<td>8.3.1. EP as a Concept</td>
<td>59</td>
</tr>
<tr>
<td>8.3.2. Good and Bad of EP</td>
<td>59</td>
</tr>
<tr>
<td>8.3.3. An Initial Validation of EP for the Customizable Generic Products</td>
<td>59</td>
</tr>
<tr>
<td>8.4. How this Report Can Benefit TAT AB and Tenk</td>
<td>59</td>
</tr>
<tr>
<td>8.5. Future Work</td>
<td>59</td>
</tr>
<tr>
<td>8.5.1. Validation of EP for Large Sized Customizable Generic Products</td>
<td>59</td>
</tr>
<tr>
<td>8.5.2. Validation in More than One Company</td>
<td>60</td>
</tr>
<tr>
<td>8.5.3. Comparing Other Approaches with EP for Customizable Generic Products</td>
<td>60</td>
</tr>
<tr>
<td>9. References</td>
<td>61</td>
</tr>
</tbody>
</table>
1. **Introduction**

The process of software development has gained much attention recently (see e.g. [2, 5, 7, 8, 9, 11]). In past, software was developed using undefined methods but the increasing need, size and complexity of software has forced researchers and practitioners towards developing and adapting defined methods for software development. Water fall model was the first step towards defining and prescribing the method for software development [2, 10]. From early 1970s to late 1980s water fall model gained lot of popularity from academia and industry and was adopted by most of the industry [10]. But water fall model could not address all the problems of industry. For example, there are many issues in market driven and customizable generic products such as requirements overload, requirements ambiguity, loose specifications, too many stakeholders, difficulty in identifying key stakeholders, volatile market, political scenarios etc [1, 3].

To overcome with the aforementioned issues researchers and practitioners have come up with different software process models like incremental development model, iterative model, agile model, rapid application development model etc [2]. But so far, no model has addressed all the problems of software industry and each has strong and weak points [10].

EP could be one of these candidates that can cope with the aforementioned types of problems and issues. In EP, an initial version of system is developed using the best known and highly prioritized requirements [3]. This initial version of the system is evolved to a final product through a number of stages by consulting with different stakeholders [3]. EP model and many other software process models are being practiced and validated by industry [9, 11], but due to time constrains, market pressure, cost and other limitations industry finds it difficult to practice these models as described in literature [7]. Industry also ignores problems aroused during practicing and validating these models. Results from practices and validation of these models may not be effective (of good quality) because it depends on many factors like organizational environment, its involvement and willingness, nature of products, and etc [1, 8].

1.1. **Gap and Its Importance**

In this thesis, we will investigate how EP is applied in industry. In literature we have found numerous case studies on EP [4, 5, 6]. But most of these are conducted either on bespoke or market driven product [4, 5, 6]. To the best of our knowledge, there is no case study conducted on EP for customizable generic product. Customizable generic products lie between bespoke and market driven products for more details see section 2.4.

Specifically, we will conduct a case study on EP in customizable generic product at the company TAT AB. TAT AB is situated in Malmo, Sweden, and develops mobile applications and user interface for mobile phones. We selected TAT AB because it is developing mobile applications and user interface. It is also facing many problems such as; short time to market, rapid change in requirement, volatile market, and too many stakeholders as described in [71]. In particular, we will develop mobile applications using evolutionary prototyping at Tenk AB which is a research and development department of TAT AB.

The main aim of this thesis is to make an initial validation of EP for customizable generic products. During this work, we will try to explore prerequisites of the selected approach, issues in the selected approach, and solutions to these issues and make an initial validation of EP in industry. We hope that the results from this investigation will help both academia and industry to gain more insight in EP. Furthermore, our study will
fill this gap by making an initial validation of evolutionary prototyping in industry for customizable generic product.

1.2. **Aims and Objectives**

The main aim of this thesis is to make an initial validation of EP for customizable generic products (mobile applications). This will be done by applying the selected approach in the development of mobile applications at Tenk. Our results can be helpful for researcher and practitioners in their decision making regarding the use of EP. In order to accomplish our aim we have to achieve following objectives:

- Perform a literature review to get an overview that what has been done in the area of prototyping and EP until now.
- To find and explain alternative approaches of EP for software development.
- In order to get reliable results of EP, there is a need to understand and study the current processes the nature and environment of the organization under consideration (i.e., TAT AB in Malmo).
- Find applicability of EP on different types of products (i.e., bespoke, market driven, and customizable generic products). Validating use of EP for customizable generic products in industry.

1.3. **Research Questions**

In order to achieve our aims and objectives we will address following research questions:

- What literature offers about prototyping and EP?
- What other alternatives are there of EP for software development?
- How EP is used by Tenk AB?
- What are strengths and weaknesses of EP for different types of products (market driven and customizable generic products)?
- Is it useful for Tenk AB to build customizable generic products using EP?

1.4. **Expected Outcomes**

The expected outcomes of our study are as follows:

- A detailed overview of prototyping and EP from literature.
- An explanation and a comparison of alternative approaches of EP.
- A discussion explaining the case study of processes practiced at TAT AB.
- A discussion of the results from the case study, together with an evaluation of the use of EP in industry.
- Customizable generic product(s) will be developed using EP in TAT AB in order to validate its results.

1.5. **Research Methodology**

Research can be classified from three different perspectives: field, approach and Nature [12]. In study we focused on area of software engineering specifically in software process models. This study involves a research conducted in industry. There are many approaches to conduct a research such as case study, experiment, action research, surveys etc [12].
We used a case study approach that is a part of qualitative research methodology in our research project. A case study involves the investigation of a particular situation. As our study also deals with the same kind of problem and that is the reason for selecting this approach. Case study is also supported by action research because this study is more focused on how practitioners do certain tasks rather by asking them. Authors of this study were involved themselves in the development process. Specifically, after the completion of each prototype data was collected based on attributes as time estimation, documentation, software testing, quality of code, quality of products, customer involvement and satisfaction, team effort, delivery time, and change management and overload. Participation of authors varied from mere observations to actual implementation (i.e. development, meetings, project planning and its analysis). The focus was to make advancement in current literature of EP for academia and industry. In this case study we investigated and evaluated EP by developing two customizable generic products using evolutionary prototyping in TAT AB.

This study is a kind of descriptive study reviewing and evaluating existed theory and knowledge in the field of software process models particularly in EP. This project involves a thorough study of EP, improving understanding in this area and identifying strengths and weaknesses of this field.

1.6. Research Settings

In order to fulfill the main aim of this study, two customizable generic products were developed using evolutionary prototyping in TAT AB namely Social Playlist and Social Location. Social Playlist was developed by six members out of which four were developers and two were from management. On the other hand, for Social location two developers and two members from management were involved. In both of these projects the main role of authors was to develop the projects, but they also participated in management team in order to collect data from both these projects regarding this study. Two days of training on Scrum was given to all the participants. The aforementioned education was provided by a specialist in scrum who is working at TAT AB. More details of research setting can be seen in section 7.1.

1.7. Methodology for Analysis of Results

Before conducting this study we studied literature resources that were dealing with same kind of problems [74, 77]. These studies helped us in deciding attributes that can be used to evaluate a software process model. In order to analyze the results from our study we used attributes such as time estimation, documentation, software testing, quality of code, quality of products, customer involvement and satisfaction, team effort, delivery time, and change management and overload.

1.8. Structure of Thesis

In section 2 we will provide definition of the concept which will be used throughout the thesis. In our thesis, section 3 will provide an overview of what has been done in the area of prototyping and EP. Section 4 will discuss different available alternatives of EP for software development process. This section will also explore and highlight strengths and weaknesses of these alternatives. In section 5 we will discuss different situations where EP could be a viable option. This section will also explain strengths and limitations of EP. Section 6 will present introduction of TAT, its processes, Tenk and its processes and applicability of EP at Tenk.

In section 7, we will present data that we got after applying EP for development of our projects at Tenk. Based on this data this section also discusses the initial validation of EP for customizable generic products. In section 8 we will conclude the research work by discussing our findings and we will also discuss future work related to thesis work in the same section.
2. **BACKGROUND**

In this chapter we will start by providing the history of software development in section 2.1 in order to discuss the need of software process models. We then continue by presenting different existing software process models in section 2.2, in particular EP in section 2.3. Finally we will describe customizable generic products in section 2.4.

2.1. **History of Software Development**

Software can be described as a set of computer programs or/and procedures, documentation and configuration that perform some task on a computer system [24]. The term “Software” was first used by John W. Tukey in 1958 [19]. The history of software began with the calculator which was the earliest version of digital computer where calculations were carried out using sequential instructions and those were software for it. In those days, hardware was considered more important than. During that era separate hardware and software was designed for different application domains that involve different tasks. But soon researchers and practitioners analyzed that this approach had lot of problems such as cost, time and complexity. To overcome these problems they introduced a new idea in which same hardware could be used to perform different tasks by only modifying software. And this resulted in the development of EDSAC (Electronic Delay Storage Automatic Computer), the first stored-program computer [22].

Advancements in the computer hardware increased the use of computer which augmented the need of software. During the initial phase of digital computer, software was developed using undefined methods but as the software needs, size and complexity increased, it became difficult to develop software product using undefined methods. This forced researchers and practitioners towards developing and adapting defined methods for software development. Water fall model was the first step towards the development of defined methods [2, 10]. From early 1970s to late 1980s water fall model gained lot of popularity from academia and industry, and it was adopted by most of the industry [10]. But water fall model could not live up to the expectations and that gave rise to new process models like Boehm’s spiral, Prototyping, Iterative & Incremental, and Agile models.

2.2. **Software Process Models**

A number of different process models for software development have been proposed, and each of these defines a set of framework activities that are conducted to organize and accomplish a software product/project. These models can be used to develop more formalized and defined description of software life cycle activities. Regardless of software process models, software engineers have traditionally selected a generic process framework with activities such as communication, planning, modeling, construction and deployment. Roger S. pressman in his book Software Engineering A Practitioner’s Approach classifies software process models as prescriptive and agile models [23]. Selection and use of software process model depends on the nature of the organization [7]. Factors such as organization’s size, the knowledge and experience of people working within the organization, hardware resources, application domain, and the corresponding software and system requirements play an important role in the selection of software process model. Here in this section we will give a brief description of these software process models.
2.2.1. Prescriptive Models

A prescriptive model describes how a software product will be developed, how software development activities will be performed and what will be the order of these activities [25]. Software models that come under this category are briefly described as:

2.2.1.1. Waterfall Model

Waterfall model sometimes also called classic life cycle model. According to Schach this model was the only model widely accepted until the early 1980 [26]. This model suggests systematic sequential approach for software development that begins with requirements gathering and progresses through planning, modeling, construction and deployment.

2.2.1.2. Incremental process Models

In market driven and customizable generic products, requirements are not clear at initial stages and it is often needed to provide limited software functionality to users quickly and then refine the software and add the functionality in later releases. In such a situation incremental model could be a better option. Models that come under this category are:

- Incremental model:
  It is an extension of the waterfall model applied in iterative way. In this model, software products are developed by small iteration or increments.

- Rapid Application Development Model (RAD):
  RAD is an incremental development model that focuses on a short development cycle and it is high speed adaptation of waterfall model.

2.2.1.3. Evolutionary Process Models

Changing of business and software product requirements over time and tight market deadlines makes it difficult to come up with end product at once. In this situation a limited version of software product could be introduced to meet market pressure and the product is evolved over time. Prototyping model, spiral model and Concurrent model are Evolutionary Process models briefly described as:

- Prototyping Model
  Prototyping model assists software engineers and end users of the product to better understand what to build when there are fuzzy requirements. A set of known and highly prioritized requirements are prototyped and are given a functional form that can be evaluated by the customer/end-user. Customer feedback is used to refine the product.

- Spiral Model
  It combines the iterative nature of prototyping with the systematic aspects of waterfall model. Software is developed in a series of evolutionary releases.

- Concurrent Model
  It can be represented as a series of framework software engineering activities, tasks and actions and their associated states. Concurrent model is applicable to any kind of software project and gives an accurate picture of its current state.

2.2.1.4. Specialized Process Models

Models that come under this category of process models are:

- Component-based Development:
  This model is evolutionary in nature incorporating many of the characteristics of spiral model and involves iterative approach to develop a software product.

- Formal Method Model:
This model consists of a set of activities that lead to mathematical specification of software project.

- **Aspect Oriented Software Development:**
  This model is often referred to as aspect oriented programming (AOP). It provides an approach for defining, specifying, designing and constructing aspects/cross cutting concern. This process helps programmers in the separation of concerns; separation of concerns involves dividing a program into distinct parts that overlap in functionality.

2.2.1.5. **Unified Process**

This model is based on approach that involves use-case driven, iterative and incremental process giving evolutionary feel to develop a software product.

2.2.2. **Agile Models**

Agile software process model aims at rapid delivery of software product by integrating light weight, incremental and iterative processes. It involves quick changes in requirements and surrounding environments [27]. There are number of agile models such as Extreme Programming, Adaptive Software Development, Scrum, Crystal, Feature Driven Development Agile Modeling [28] and these are defined in the following sub sections.

2.2.2.1. **Extreme Programming (XP)**

It involves object oriented approach to develop a software product. XP consists of four framework activities: planning, design, coding and testing. XP suggests a number of powerful and innovative techniques that help an agile team to build frequent software releases.

2.2.2.2. **Adaptive Software Development (ASD)**

ASD is used to build complex software systems [29]. It focuses on human collaboration and self organization of team. It involves three framework activities: speculation, collaboration and learning.

2.2.2.3. **Dynamic System Development Method (DSDM)**

DSDM is an iterative software process that provides a framework for creating maintaining software projects that have tight time constraints. It involves three iterative cycles: functional model iteration, design and build iteration and implementation.

2.2.2.4. **Scrum**

Scrum defines a set of principles that are used to guide development activities in a process of projects with tight time lines, changing requirements and business criticality. These framework activities include requirements, analysis, design, evolution and delivery and each activity is done through a process pattern called sprint. Scrum meetings are conducted on daily bases that are very short (normally 15 minutes) [30].

2.2.2.5. **Crystal**

Crystal is an agile process that is very helpful for different types of projects with different sizes and complexity. Crystal process also follows the iterative strategy to build a software product.

2.2.2.6. **Feature Driven Development (FDD)**

In FDD, a feature is a function that is more valuable to client and can be implemented in two weeks or less. FDD has more focus on project and quality management than any other agile method [23].
2.3. **EP**

In order to understand EP, first we need to understand what prototyping is. IEEE defines prototyping as "*A type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process*" [33, 32].

In other words, we can say that a prototype is an executable form of a system that reflects a selected subset of its properties. Prototypes help in formulation and validation of requirements, resolving technical design issues, and supporting computer-aided design of both software and hardware components of proposed systems.

The selected approach produces a series of prototypes in which the final version becomes the software product [73]. EP model is very useful for the development of customizable generic products and market driven product. As EP involves development of product in iterations, organization can get idea about the market's/customers' response to the product in the initial iteration of product/application development. If company finds that the product is not of market's/customers' interest, it can stop its development that can save the cost of developing complete product.

2.4. **Customizable Generic Products**

Software products can be classified in three categories namely bespoke, market driven and customizable generic product. As we know that bespoke products are only developed for a specific customer and market driven products are developed to target a market of people. On the other hand customizable generic products are developed to target a market with a specific interest. Customizable generic product can also be defined as software which contains a generic application framework that can be tailored to the client’s requirements [82]. In other words we can also say that customizable generic products lie between bespoke and market driven products. Software companies have realized that, although their clients have different needs, but there are also similarities in the needs of different customers. Different customers with almost similar needs can be tackled with customizable generic products. CRM (Customer Relationship Management) is a good example of customizable generic product. As it is unique for each company but it also contains many common features. So we can design a generic application framework out of common features and it can then be tailored for any client by integrating features demanded by a specific customer.

Customizable generic product can be developed by different methodologies. Barm and Becker [81] have discussed component based and re-use oriented software development approach in the domain of embedded operating systems. They have classified software component in two categories: executable component (application or software system) and generic software components that are used to build application at the program construction time. They have also identified that beneath the surface, different customers share some common requirements and that can be transformed into a generic framework. In this way we can customize the developed application for a customer by adding new components based on the client’s requirement and needs.
We have also developed a generic framework for both of our applications developed at Tenk. It can then be customized for any customer by adding components based on the client’s requirement and needs. More details about what components can be added and removed into the generic framework of applications developed at Tenk during the research are discussed in section 7.1.
3. **OVERVIEW OF PROTOTYPING AND EP**

In this chapter we will study existing literature on prototyping and EP. This literature study will enable us to know more about EP that will support us in applying it in our projects at Tenk more effectively. This literature review will also help us in achieving main aim of our thesis that is to make an initial validation of EP for customizable generic products.

Today many models exist for the software development process such as waterfall model, iterative/incremental, prototyping, agile etc. But here we will only focus on prototyping and EP. In this chapter we will present some existing literature on prototyping and EP from 1991 up to now. We studied literature in different periods for following reasons:

- To find state of art in the area of prototyping
- What changes are brought by the prototyping in software industry
- Analyzing impact of prototyping on software industry in different periods
- To find evolvement in the area of prototyping

Prototyping started to get focus from the academia and industry of software development in the late 1980’s [54], but it is beyond the scope of our thesis to discuss literature from 1980. Therefore, we will present literature from 1991 till now. Rest of the chapter is organized in different sections. Section 3.1 to 3.4 will present existing literature on prototyping in four different periods. In section 3.5, we will conclude this chapter.

3.1. **Prototyping from the Period of 1991-1995**

In this section, we will present literature on prototyping in the area of software development from 1991-1995. For this section, we have selected 10 articles on prototyping from 1991 to 1995 and at least one literature resource is selected from each year.

During our study of literature from 1991-1995, we found that literature was targeting the following areas of prototyping:

- Methods, Approach and tools for prototyping (Taxonomy)
- How to apply prototyping in different phases of Software Development Life Cycle (SDLC)
- Evaluation of prototyping technique
- Pitfalls of EP
- Future benefits for the company using prototyping

3.1.1. **Methods, Tool and Approach for Prototyping**

In 1990’s concept of prototyping oriented development was very new for software industry but it was successfully applied in other traditional fields of engineering such as civil engineering, mechanical engineering and etc. The results of prototyping in these fields were very effective and that changed minds of the people who were working in software industry.

Until 1990 there was no agreement on methods and tools for supporting prototyping oriented development in software industry and there was confusion in the terms of prototype and prototyping. But literature from
early 1990’s suggested that researchers were reached to some consensus on the definition of prototype and prototyping. In [65, 70], prototype is defined as “a dynamic visual model providing a communication tool for customer and developer that is far more effective than either narrative prose or static visual models for portraying functionality”. Prototyping can be defined as “a specific strategy for performing requirements definitions wherein user needs are extracted, presented, and successively refined by building a working model of the ultimate system quickly and in its working context” [70, 64]. According to Pomberger et al [70] phase oriented development model and prototyping oriented model complement each other. The major difference between prototyping oriented software development and phase oriented software development are deliverables, iterative cycle, requirements. In prototyping deliverables are produced at each phase on the other hand in phase oriented development deliverable is produced at the final stage of development. In prototyping oriented development, implementation can be performed without having clear requirements, which means that implementation could be started early as compared to the waterfall model. In [70] they did not make any distinction between iterative and prototype development.

Different methods exist for prototyping such as visual specification, traditional graphical approaches, textural specification language, programming language, storyboarding and scenarios, object oriented paradigm, rapid programming, software reuse and software storming [43]. Asur and Hufnagel [43] also discuss different tools for prototyping like computer aided prototyping system (CAPS), programmable network prototyping system (PNPS), QUISAP, EPROS and etc.

It is clear from this section that prototyping started to get focus in 1990 from software industry and academia and we find some starting point literature on prototyping in this era [70, 42, 43].

3.1.2. How Prototyping Was Applied in Software Development Model

In the beginning of prototyping, many researchers and practitioners think that prototyping could not be applied completely as a software development model [70, 41, 43]. So in the beginning it was applied with traditional phase oriented development model.

We found that most of the selected literature was discussing rapid prototyping. Rapid prototyping can be divided in to two categories namely throwaway and EP. But there is another type called operational prototyping which is a combination of throwaway and EP [41]. Acosta et al [38] explain the applicability of prototyping techniques in requirements engineering environment (REE). According to them prototyping techniques are very helpful in gathering requirements when you have some kind of prototype for the original product.

REE was presented by Rome Laboratory and it is an integrated set of tools that facilitate rapid development functions, user interface, and performance prototype model of system component. Prototyping is a suitable candidate for development when requirement are unclear or ambiguous. In prototyping requirements become clear in the later phases of development. And debugging/testing phase started very soon in prototyping oriented development as compared to phase oriented development.

Asur and Hufnagel [43] defined prototyping process by steps as follows:

- Gathering requirements from customers in form of sketches, use case scenarios, and terminology
- Designing specification with the help of prototype description language
• After designing phase implementation is performed. Unlike of phase oriented development approach, output of implementation phase is used as input for further analysis.
• Then step of the prototype evaluation is executed with the help of customer, developer and end user. The output of this step is also used as an input to the requirements step.
• Prototypes are refined iteratively at this stage and it is necessary to make changes as soon as possible so that updates can be reevaluated.
• Requirements are modified and updated according to the feedback from previous step.
• In final step production system is designed and implemented.

3.1.3. Analysis of Prototyping in Industry

It was very interesting to find that prototyping was introduced in late 1980’s in software industry and numerous case studies on evaluation of prototype had been published in early 1990’s. This section will present evaluation results of prototyping from different selected literature. The selected literature discusses evaluation of prototyping from three different perspectives:

3.1.3.1. Evaluating prototyping by different attributes

Here we will discuss three attributes i.e. product attributes, process attributes, and problems from [36].

- Product Attributes
  Product attributes include ease of use, user needs, and number of features, performance, design quality and maintainability [36]. According to Gorden and Bieman [36], overall product attributes shows positive trends towards the EP except performance, design quality and maintainability. In [36], they also concluded that EP can also be used for large products.

- Process Attributes
  Process attributes comprises of effort, cost estimation, end user participation and expertise required [36]. Experience and skills of developer plays a very important role in prototyping oriented development. It was also found that cost for the development was reduced by prototyping [36]. In some cases prototyping oriented development helps in good estimation of cost but in some cases it is contradictory. Another factor is user participation which is increased a lot because user feels more comfortable with prototype as compared to requirements specification. And sometimes too much user involvement takes the form of political maneuver.

- Problems
  Some problem that are identified in case studies are product performance, misunderstanding by end user, code maintainability, delivering a throwaway, budgeting, completion and conversion, and handling development of large systems.

3.1.3.2. Factors Given no importance in the Evaluation of Prototyping

Some new factors which were identified in case study [39] have not been given any importance in the earlier case studies conducted for the evaluation of prototyping. These factors include:

- Knowledge of client about prototyping
- Knowledge of the application area
- Inappropriate documentation
- Participation of End user
- Expecting too much from user
- Prototyping is learning
- Face to face communication is preferred
- All development activities shall be define in a written form
- Neglected documentation
- Presentation prototype have been under estimated
- Reusable component increases creativity
- Prototyping is more than user interface

Further information about these factors could be viewed in [39], here only the main headings are mentioned.

3.1.3.3. Evaluating Effect of Prototyping on Software Quality

Software quality can be categorized in two categories that is external and internal quality. External quality consists of ease of use, match with user need, feature, and maintainability while internal quality includes design quality such as lines of code and etc. It was mentioned in [37] that rapid prototyping has positive impact on the external quality while it has some negative impact on the internal quality.

3.1.4. Pitfalls of EP

It is important for developers to address performance issues at early stage in the process when they are developing product with EP. This is because the parts of the prototype are later to be included in the final system [36]. The main drawbacks with EP are poor design quality and maintainability, underbidding, and misunderstanding between developers and users. According to [37], quality attributes such as performance, design quality, and maintainability can suffer during EP if some proper steps and standards are not enforced to avoid the related pitfalls. Some other problems with EP could be the need of skilled and experienced programmers and maintaining problems due to difficulty in maintaining the code [37].

3.1.5. Future Benefits for Company Using Prototyping

Wohlers [40] identified some future benefits for companies that were using prototyping. Some of future potential are given below:

3.1.5.1. Awareness

Many companies were unaware of potential benefits of prototyping but soon they realized benefits of rapid prototyping (RP). Wohlers mentioned that extensive research was conducted in the area of rapid prototyping and companies can use this as a tool to aware peoples working in company.

3.1.5.2. Erasing Myths

RP is erasing many myths of companies such as RP products are not stable, having low quality standards and fragile. This will lead to introduce more RP products and parts in future.

3.1.5.3. Technical Enhancement
Companies that are developing products using prototyping can make technical enhancements even after product is deployed and ready to use. This will increase sales of a company and it also involves customers’ feedback in technical enhancements.

3.1.5.4. Personnel and Organizational Issues

RP oriented development changes the attitude of organizational personnel into more positive way. This helps organization to grow faster and in right direction.

3.1.5.5. Complementary Process and Technologies

Complementary process and technologies that will be used with rapid prototyping helps company in assessing risks, future use of product, future problems and etc. These tools provide very good data on which organization can make their future decisions.

3.2. Prototyping from the Period of 1996-1999

In this section we will discuss prototyping practices in the industries discussed in literature from 1996 to 1999. And same as above we tried to study at least one article from each year so that the current topic could be covered. Literature review shows that the prototyping became more and more popular as the time passed. Here we will discuss prototyping in following sections:

- Approach
- Comparing EP with Waterfall and Agile
- Benefits
- Drawbacks
- Practices in industry
- Prototyping user interfaces
- Evolution

3.2.1. Approach

EP could be used in the development of large and complex software systems in iterations with high performance and low cost. Prototyping is a development approach used to improve planning and execution of software projects by developing executable software components (prototypes) for experimental purpose [57]. Prototyping is very helpful to extract experience from new application areas and for supporting evolutionary software development [57]. The success of EP is based on careful selection of the tools and techniques that provide fast iteration process. EP approach was considered as suitable for small highly motivated team with great skills and qualification [52]. Most of the processes, methods and approach were almost the same as described in the earlier section.

3.2.2. Comparing EP with Waterfall and Agile

The reason to compare EP with Waterfall and Agile is that Waterfall is quite traditional approach for software development and Agile is the latest development approach for software development and also got very much attention from the industry in the selected period of time. In Waterfall model each phase is completed before the next is started and requires a complete set of clear requirements in order to start
design and implementation of the system. On the other hand in EP, it is not necessary to complete the previous phase for entering in the next phase and the first iteration can be started with the known set of requirements. Verner and Cerpa [56] suggest that EP provides better communication between development team and end users, problem detection at early stages, more flexible designs and high quality system with less cost than a waterfall approach. Further Verner and Cerpa [68] did a research with many software organizations and from their findings they suggested that prototyping approach was not easier to learn to use than a waterfall approach, and observed that systems developed using prototyping did not have less functionality than systems developed with a waterfall approach. As already described in section 2.2, that agile software process models aim at rapid delivery of software product by integrating light weight, incremental and iterative processes. It involves quick changes in requirements and surrounding environments. Aoyama [27] suggested that the Agile software process is based on incremental-delivery and evolutionary model by which products are incrementally delivered over time. Both in Agile and EP, customer involvement is very high through the development process. In Agile after each iteration a working piece of code is delivered, while in EP a function prototype is given after completion of each iteration. In both the methodologies software product is evolved through a series of prototypes/working piece of code. Further comparison of EP with Agile can be seen in section 4.2.

3.2.3. Benefits of EP

In 1999 Michael Rees suggests that rapid prototyping is the most powerful and flexible tool that has ever been invented [53]. Study of the selected literature showed that EP process for software development has many advantages as:

- **Communication with Users**
  Prototyping provides better communication between the development team and end user and allows users to know more about the system before its deliver [56]. User can request changes in the proposed system and feel closer to software development process.

- **Design Flexibility**
  Prototyping gives design flexibility during the development process, especially when there are unclear requirements or when the developers are inexperienced with the system being developed [56]. Developers can start by implementing the functions that are more clear and easy to implement [52].

- **Discovery of Problems at Early Stages.**
  First prototype of the of the system could be started with the initial known requirements[51]. By releasing the prototype of the initial known requirements to the end user helps the end user to explain the developer what they really need. This helps the development team in finding problems at early stages of software development [66, 56, 67].

- **Achieving Intended Functionality**
  Prototyping involves end user in each iteration of the system [55]. End users can suggest some improvements and changes that suit their needs. This helps in the delivery of final system with full intended functionality.

- **Easy Adaptation**
Since the Evolutionary model is very close to the “just do it” approach so for the companies that are not following any of the process models it is easy to adopt Evolutionary rather the more strict models like waterfall approach [52]

- **Product Validation**
  With prototyping approach validation of the product was quite easy and was performed during prototyping delivery. Each time when a prototype completed it could be evaluated and validated by the end user and the final system developed by the combination of such prototypes was ultimately a validated system.

### 3.2.4. Drawbacks

From the studied literature, where the use of rapid prototyping was enriched with many of benefits/advantages, there it also had some drawbacks in its use as:

- **Structure of the System**
  The problem anticipated with evolutionary approach was the classical bad structure of the application produced under iterative process [52].

- **Project Control**
  Prototyping approach was not good in managing schedule, cost and the project as whole as others processes like waterfall did [56]. It was also difficult to measure the process evolution and progress towards the final solution. [52]

- **Maintainability of Produced System.**
  It has been seen that industries using rapid prototyping as development process were feeling difficulty in maintainability of the final system [67].

- **Product Verification**
  Another key issue with EP is the verification of the product. Verification of the system can only be performed when we have well-defined system specifications and that is not the case in rapid prototyping.

- **Language Dependency**
  Since prototypes depend on implementation language therefore these cannot be reused across different operating environment [51].

### 3.2.5. Practices in Industry

Selected literature shows that EP was being used by many of the software companies. Verner and Cerpa made a survey of 82 companies from Australia and Hong Kong (43 from Australia and 39 from Hong Kong) who were following different software development processes. In their study they suggested that small companies preferred a prototyping approach, medium sized companies preferred a waterfall approach, while large companies were more aware of the benefits of both approaches [56]. Jardim and Cunha did a case on a software engineering project called SITINA (Sistema de Telemetria para Instalações Não Acompanhadas) that was built using EP at Portuguese, Small to Medium sized Enterprise Portugal. They describe that the motivation behind selection of prototyping was the experience of development team with 4th generation tools and the ability of prototyping to tackle badly defined requirements. The project took
more iterations than they could expect. Company continuously involved end users for requirement gathering and prototype evaluation. Case study suggests that communication between the development team and the end users was a key factor for project success [52].

Dr. Gaubatz [55] discussed rapid prototyping practices at Delta Clipper Programs. The company was using rapid prototyping to develop its products since its beginning. Company had written guidelines for rapid prototyping. These guidelines were signed-off by the senior management of the major functional departments and were periodically updated to reflect project changes and to incorporate lessons learned. Prototyping team was developing the prototypes within defined and tight schedule and each team was responsible for delivering a product/service that met agreed requirements. The team consisted of members from all disciplines like analysis, design, test and management. Company involved customer as an essential participant to help the team in design and test of the product and to reduce the complexity of requirements. At Delta Clipper management team was performing inspiring and leading role for the team to give high performance in their work. Study of Delta Clipper Programs tells that Prototyping environment enabled small teams to develop and deliver products with high quality in short time with low costs. Study also shows that DC-X project at Delta Clipper was completed using rapid prototyping and this process resulted high quality product in two years with the cost less than the half that had been expected/planned for a traditional program approach.

Stanley and Chatterjee [51] suggested knowledge based modeling approach to EP. Each evolving prototype of the system could be though as an executable model of the target system and could be executed to test its functionality and performance at any evolution stage. This prototype could be started based on the functions that the programmer believed implemented easily. If a prototype was failed to assure the intended requirements, it was then modified and improved. During the entire EP, KBM could be used to store metadata and information related to the developed software system. This model helped the programmer to decompose the software components. This model also facilitated the programmer by enabling him to write either real or simulated code for the functions to be implemented.

3.2.6. Prototyping User Interfaces

From the past few years the development of highly interactive software systems with graphical user interfaces has become very much common. Therefore, acceptance of such interactive systems depends on the quality of their user interfaces. Baumer et al [57] did a case study on nice major industrial projects where the main focus was on user interface prototyping using different tools to develop user interface prototypes. They concluded that prototyping is an excellent mean for generating ideas about how a user interface could be designed. They also suggested that user interface prototyping helps in evaluating the quality of a solution at an early stage and that was the main reason that why the prototyping was applied in an increasing number of projects. Authors classified user interface prototypes in four different types:

- Presentation prototypes are developed to demonstrate how an application may solve the given requirements.
- Functional prototypes are built to illustrate both the user interfaces and the functionality of the planned application.
- Breadboard prototypes help in investigating technical aspects and their associated risks such as architecture or functionality of the planned system.
- Pilot prototypes are considered very mature prototypes and could be practically applied.
3.2.7. Evolution till 1999

From the literature we can see that researchers were more focused towards defining processes and methodologies, standardizing approach and finding tools in order to apply prototyping in different phases of SDLC in a more suitable way. Literature was also more focused towards the evaluation of different techniques for prototyping model and looking for its future benefits. But from 1995 to 1999 the trend was changed and prototyping was being used by most of the software industry as a software development model. 1996 was the strong year for the rapid prototyping [54]. We found some case studies by different researchers describing the uses of prototyping in software industry and discussing its advantages and disadvantages for the applied projects. In this era we found that researchers were more focused on the application of prototyping in industry.

3.3. Prototyping from the Period of 2000 to 2004

For this period we have selected ten research papers and most of them were discussing agile methodology with prototyping. We also included a research paper which discusses process diversity in SDLC. We will discuss this section from the following perspectives:

- Why process differs in software development
- Agile methodology and prototyping
- Prototyping and Requirements Engineering
- Prototyping and Software Testing
- Prototyping and Web Application
- Evolution

3.3.1. Why process Differs in Software Development

Today software industry demands a process that is more adaptive in nature. Numerous software process models exist that are adaptive in nature such as iterative and incremental, agile and prototyping.

There is no universal standard approach by which software could be developed; rather every organization has its own process for software development based on their needs, application domain and nature of the organization. So far, many software process models have been introduced for software development but none of these fulfills all the requirements of software industry. It is very difficult to point a software process model that fulfills all the needs of software industry [7]. Therefore, it becomes very important for a company to find process model that suits to its needs, environment and business strategy [7]. In present environment companies are facing fierce competition and market is very volatile.

3.3.2. Agile Methodology and Prototyping

The adaptive nature is a core essence of agile methodology. Agile methods can easily adapt changes occurred at different times of development process. In our literature study we found many case studies that were relating EP with agile prototyping. Many research papers were comparing EP with XP [60, 61, 62]. EP got a lot of popularity after the introduction of XP. EP can be compared with different agile methodology such as XP, crystal methodologies and adaptive software development as described by [61]. In XP after
each cycle a well designed and tested code is produced, while in prototyping the major focus is on functionality not on the quality of code. Dagnino introduced a new approach called ADEPT in 2002 [60]. ADEPT stands for agile development in EP technique, ADEPT is not a completely new approach rather it contains practices from XP, Scrum and DSDM. ADEPT was developed to achieve following set of objectives:

- Provide an iterative and incremental approach to developers so that they can develop product more rapidly.
- Provide mechanism that could adapt to changing requirements.
- Minimize documentation so that developers can give more focus on the working piece of software
- Provide inspection for the management and control purpose.
- Increase the speed of SDLC.

The areas of development that are common in ADEPT and Agile are; process, risk reduction, focus on working software, adaptive emergent and self organizing work and team, customer interaction, iterative and incremental development and management practices. Another point of view from researches is that Agile models are extended form of iterative, evolutionary and incremental development [60, 61, 62]. Roots of these approaches are very old, IBM started incremental development in 1957. It was in late 1980 when EP was commonly used for the development of AI systems with the help from iterative and incremental practices. And in 1987 EP was used by incorporating methods Cleanroom. Larman and Basili [63] suggested that different methods were used with EP in different eras.

3.3.3. Prototyping and Requirements Engineering

Prototyping has got great acceptance in the software engineering community as a reliable model for software creation. Increasing costs of software and the decreasing rate of software systems that meet customer needs has made EP as a viable software development model. It addresses a number of problems like numerous maintenance requests, premature freezing of requirements, difficulty in revising previous model stages, lack of consideration of user-system interfaces and miscommunication between developers and customers that are not sufficiently addressed in more traditional software process models. EP is suitable for gathering a correct and consistent set of requirements. This approach is useful for building quality software through clarification of existing requirements and the discovery of previously missing and unknown requirements. Carter, R.A. et al [3] proposed a model called EPRAM (EP with Risk Analysis and Mitigation). This model was addressing the challenges inherent in rapid development efforts in electronic commerce. Designers of this model assure that it facilitates the construction of a complete and consistent set of system requirements and mitigates the poorly affected requirements creep by supporting early detection and resolution of requirements conflict.

3.3.4. Prototyping and Software Testing

In iterative and incremental model, it is very necessary to know the direction of development. And this is a major drawback of incremental and evolutionary development that sometimes it becomes difficult to decide the direction of development. This problem could be resolved by reference architecture. Theses reference architecture models can be used with code to test the application and one such kind of approach is proposed.
by Pretschner et al [69]. Pretschner et al [69] discussed the use of executable models in EP. In [69] they also mentioned that testing process can start much earlier and can be more affective, if executables models are used in EP [69].

3.3.5. Prototyping and Web Applications

Chen and Huang [59] applied prototyping model for the development of stream based lecturing system. This system was developed in three iterations by using EP. First version of the system was named as synchronization mode, second was named as browser capture mode and third version was called full screen capture mode. According to authors the motivation behind the selection of evolutionary approach was unclear and incomplete user needs. There was no discussion on evaluation results of EP approach that was used for the system development.

Bleek et al [58] presented a modified approach of prototyping called e-prototyping for the development of web applications. The main reasons for the introduction of e-prototyping are:

- In web projects requirement gathering is different from desktop projects and products.
- There are many relevant actors and their perspective should be included to save expenses and foster innovation.
- The need for short iterations for web development.
- Active communication channel between users and developers is needed to exchange information.

There are four steps involved in e-prototyping functional selection, construction, evaluation, and decisions on the further use.

Another approach for development web application with the help of prototyping is WARP (web application rapid prototyping) proposed by Bochicchio and Fiore [6]. WARP provides an environment to develop and design web applications with prototyping. In WARP web application is developed in five steps: initial phase, second phase, third phase, fourth phase and last phase.

3.3.6. Evolution till 2004

In this era, we found many research articles and papers that were discussing relationship between agile and EP. From the literature it is evident that Agile methodology is an extended form of iterative, incremental and evolutionary software process models. This is the big evolution in the area of prototyping and presently many companies are developing software using Agile methodology. During this era researches also discuss the applicability of prototyping oriented development in different domains. In this regard specific environment that contains tools and methods for applying prototyping oriented development had been introduced. These tools and methods are being used by software industry today as well.

3.4. Prototyping from the Period of 2005 to Onward

This section discusses literature on prototyping from 2005 till now. In this era we found that the research was more focused towards the development of new tools and techniques for prototyping. We also found some studies that were discussing how these tools could benefit prototyping oriented software development. This section will be discussed by the following perspectives:

- Prototyping in SDLC
• Early acceptance testing of mobile applications using rapid prototyping framework.
• Prototyping for web applications and Artificial Intelligence applications
• Use of prototyping for user interface acceptance testing
• Evolution

3.4.1. Prototyping in SDLC

Prototypes can also be used as a source to convey and idea to others. There are four purposes of prototypes [48]: proof of concept, proof of product, proof of process and proof of production.

We found two research papers discussing how the different activities of SDLC could be performed with prototyping [35, 48]. As mentioned in [48] prototyping is being used by almost all the software industry intentionally or unintentionally. The introduction of web based technology stressed a lot on prototyping oriented development. The methods for software development can be categorized in two classes: specification based and prototyping based. In specification based methodology the focus is on defining clear and complete set of requirements. But even in the specification based methodologies software industry is using prototyping unintentionally. We know that SDLC consists of four main phases: Analysis, design, implementation and testing. In specification based methodology analysis is performed in two steps: planning and analysis. For planning phase this methodology uses demonstration prototypes that are developed to visualize user interface for the proposed systems and for the analysis phase discovery prototypes are used in requirement gathering. Memmell et al [45] introduced a method of model driven prototypes for corporate software specifications. Traditional paper based methods to gather requirements create communication problem between the customer and the developing organization. Thomas et al suggested that this problem could be resolved by using model driven prototyping for corporate software specification. This methodology uses user interface prototypes in design phase of SDLC. In implementation phase this methodology uses feasibility/performance prototypes [35]. All the prototypes used by this methodology in different SDLC phases are throw away prototypes.

Evolutionary development is the essence of prototyping based methodologies. In this methodology prototypes are evolved into final system. The main types of this methodology are: Boehm’s spiral model and rapid application development.

The prototyping oriented development reduces design risk involved in full production. Prototyping can also be used to compare different available alternatives for design phase.

Time and skills are the prerequisite for prototyping. Prototyping in design phase involves an overhead of time and human recourse.

3.4.2. Early acceptance Testing of Mobile Applications Using Rapid Prototyping Framework

Today mobile software industry is facing a lot of challenges such as competition, market volatility, short time to market and cost [44]. To overcome these problems Schwaiger et al [44] introduced rapid prototyping software framework for early acceptance testing of mobiles. Mobile market is growing exponentially and this brought fierce competition in mobile industry. This leads to a high risk of miss
investments if a wrong application is developed for mobiles. W. Schwaiger et al [44] recommended that agile software development should be used in the development of mobile application. Rapid prototyping can be achieved by using agile development methods. Product developed using this approach involves different iterations that enables companies to make an early decision whether the product will be accepted in market or not. The company can save cost and effort in developing a product that will not be accepted in market. There are different approaches for acceptance testing as described in [44]: Wizard of Oz, paper and voice mail based diary prototyping, experience prototyping and mobile scenario presenter. The framework proposed by W. Schwaiger et al [44] is comprised of six steps. In first step customer initiates project or a new idea is generate by the company itself. Based on requirement analysis is performed in step two. In step three, prototypes and simulation for the requirements are developed. Step four performs collective walk through and helps in deciding whether the results are according to need of user or not. If results are not according to user’s need then the process again starts from step two. But if the results are satisfying then step five is performed. At step five acceptance analyses is performed in real context and it also involves decision making regarding application acceptance. If application is not accepted, it will be rejected. But if application is accepted then step six is executed. In step six implementation and deployment is performed.

3.4.3. Prototyping for Web Applications and Computer Applications

Presently the complexity and features of web applications are growing. It is very difficult to incorporate all these functionalities at once. Since prototyping aims to address complex and big problems in iterations so for web applications prototyping provides cost and time affective solutions. There are many factors in the web application development that are not being addressed properly in conventional development practices. These factors may include client uncertainty, difficulty in requirement gathering, too many end users, vision driven in nature instead of requirements driven etc. In these situations researches suggest that prototyping should be used for web application development [46]. C.C. Lim developed web authoring toolkit for e-learning portals using middle ware technology [46]. This toolkit helps web application developers to rapidly produce prototypes with minimum amount of efforts. This toolkit also helps organizations to deliver functional e-learning portal with less amount of time and effort. Nunes and Schwabe [50] introduced hyper DE environment. This environment is a combination of model driven design and domain specific language to facilitate rapid prototyping for web applications. This grouping helps designers and developers in writing code by manipulating application models. It also facilitates to make changes in application models at runtime. Mørch [49] used EP to develop integrated software agents. Motivation behind selecting EP was that the authors did not have clear idea or clue about what they want from a system. Therefore, they started an experimental simulation known as Wizard of OZ studies. This technique is very effective in terms of cost and time for testing a new idea. With the help of this technique authors developed a realistic simulation in very short time and effort as compared to programming languages.

3.4.4. Use of Prototyping for User Interface Acceptance Testing

Due to different input and output constraints, user interfaces need to be different for each device. Designers have to design user interface for each device themselves or use a program that automatically creates user interface. Lin and Landay [47] created a prototyping tool Damask, which targets web user interface that runs on pc and mobile phone. Damask helps designers to sketch the design for a device using design pattern
to give higher level concepts within their design. Damask generates user interfaces for other devices automatically that could be refined by the designers if necessary. Authors performed a study on twelve user interface designers and found that the user interfaces created with Damask using patterns and layers took less time and were better than user interfaces created without patterns and layers.

3.4.5. Evolution till Now

From the selected literature we found that the methodology of prototyping has evolved from simple iterative and incremental development to modern day agile methodology. It was also very interesting to found that intentionally or unintentionally prototyping methodology is being used by almost everyone. Now many researchers come to the consensus that prototyping oriented development can handle challenges faced by the software industry of today such as dynamic nature of market, requirements’ change, short time to market, and customer satisfaction and accepted as an adaptive development model. We observed that activities present in prototyping oriented development are being defined and more formalized [3, 12, 60, 61, 62]. More and more tools for prototyping oriented development are being introduced, precisely for the user interface and web applications [44, 46, 47, 49, 50]. The use of prototyping oriented development is increasing day by day and presently most of the industry is using this approach to solve more complex problems.

3.5. Conclusions

The main theme of this chapter is to explore what literature offers in the area of prototyping. During our literature review we found that prototyping started to get focus from the researchers and practitioners of software world in late 1980’s. But most of its taxonomy was defined in early 1990’s and during the same period we also found some studies that were discussing the evaluation of prototyping in software industry. From mid 1990 to late 1990 we saw that more research was conducted on the application of prototyping as a software development model. The period from 2000 to 2004 has witnessed to the introduction of many agile methodologies that were based on the prototyping and iterative/incremental development models. From 2005 till now many special tools are being introduced to supports prototyping oriented development. It is evident from the research of 2005 and onwards that now prototyping is being used by most of the industry.
4. ALTERNATIVES OF EP FOR SOFTWARE DEVELOPMENT

As mentioned in Section 2.2, software process models can be classified into two categories i.e. Prescriptive and Agile. In this chapter we will compare different models of Prescriptive and Agile (adaptive) to explore different alternatives that are available for software development. This chapter also presents rationale for the selection of EP as a development for this study.

4.1. Alternatives of Evolutionary Process Model for Software Development and their Comparison

During our work at TAT and from our literature review we found that agile methodology is based on evolutionary and iterative/incremental model [27]. According to our point of view agile is the extension of EP. We used six attributes for the comparison of software development models and these are: Development Cycle, Flexibility, Risk Management, Change, Customer Involvement, and Deliverables to the Customer. We compared two attributes in each table.

4.1.1. Prescriptive Models

Table 4.1.1.1 Comparison of Prescriptive models with respect to Development cycle and Flexibility [23, 24]

<table>
<thead>
<tr>
<th>Models</th>
<th>Development Cycle</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Water Fall Model</td>
<td>Development is carried out in a large cycle and if problem occurs then whole process starts from the beginning.</td>
<td>There is no flexibility in waterfall model to move backward or forward in other phases because flow is sequential in waterfall model.</td>
</tr>
<tr>
<td>Incremental Process Models</td>
<td>Small Development cycle is used in Incremental Process Models.</td>
<td>There is flexibility to move backward or forward in other phases.</td>
</tr>
<tr>
<td>Evolutionary Process Models</td>
<td>Development is carried out in a series of evolutionary releases.</td>
<td>Evolutionary process models also supports movement of backward or forward in other phases.</td>
</tr>
<tr>
<td>Specialized Process Models</td>
<td>Most of the models in this category follow the same development cycle of Evolutionary process models.</td>
<td>Most models from Specialized Process Model support same flexibility as of Evolutionary process models.</td>
</tr>
<tr>
<td>The Unified Process</td>
<td>Development cycle is similar to incremental process models.</td>
<td>Same flexibility as of incremental process models.</td>
</tr>
</tbody>
</table>

Table 4.1.1.2 Comparison of Prescriptive Models with respect to Risk Management and Change [23, 24, 21, 11, 20]

<table>
<thead>
<tr>
<th>Models</th>
<th>Risk Management</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Water Fall Model</td>
<td>Waterfall model does not provide</td>
<td>Changes at any stage will</td>
</tr>
</tbody>
</table>
any formal mean for the risk management

<table>
<thead>
<tr>
<th>Incremental Process Models</th>
<th>Risks are handled through critical and non critical threads [20].</th>
<th>As the development of project is carried out in different iterations, therefore changes can be addressed in the next iteration of project’s lifecycle.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolutionary Process Models</td>
<td>Spiral model explicitly incorporates the activity of risk management. And in other models of evolutionary process models, risk is mitigated or handled by developing product in small iterations.</td>
<td>Any unknown changes could be addressed in the next iteration of project life cycle.</td>
</tr>
<tr>
<td>Specialized Process Models</td>
<td>Formal process model manages risk by mathematical modeling, while other models of this category need assistance from the other risk management processes.</td>
<td>Formal model can address small amount of changes but if the changes are significant then experience and lessons learned from it are retained and model is discarded [21]. In component-based development process, changes are handled at different stages of project life cycle [11].</td>
</tr>
<tr>
<td>The Unified Process</td>
<td>The unified process involves development in iterations, iteration with greatest risks are addressed first.</td>
<td>Changes are managed through iterations.</td>
</tr>
</tbody>
</table>

Table 4.1.1.3. Comparison of Prescriptive Models with respect to Customer Involvement and Deliverables to the Customer [23, 24, 12, 13]

<table>
<thead>
<tr>
<th>Models</th>
<th>Customer Involvement</th>
<th>Deliverables to the customer</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Water Fall Model</td>
<td>Customer is only involved at the initial step of waterfall model known as Requirement Analysis.</td>
<td>The water fall model produces a complete final product which is delivered to the customer</td>
</tr>
<tr>
<td>Incremental Process Models</td>
<td>Interaction with customer takes place after each increment [12].</td>
<td>Client is handed over with partial implementation of a total system after each increment.</td>
</tr>
<tr>
<td>Evolutionary Process Models</td>
<td>During Evolutionary process models, regular meetings and communications are arranged with the customers.</td>
<td>Each iteration ends with a running prototype that is given to customer. This process continues until the development of final product is not finalized or stopped.</td>
</tr>
</tbody>
</table>
**Specialized Process Models**

It is very difficult for customer to understand the formal process of software development, because it involves complex mathematics.

Specialized process models are similar to incremental models in this regard.

**The Unified Process**

The unified process is very flexible for the customer involvement. It is up to customer to decide the level of involvement.

In unified process, each iteration ends up with an executable release [13].

### 4.1.2. Agile Models

Table 4.1.2.1 Comparison of Agile models with respect to Development cycle and Flexibility [23, 24, 14, 15, 16, 17, 18]

<table>
<thead>
<tr>
<th>Models/ Attributes for Comparison</th>
<th>Development Lifecycle</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme Programming (XP)</td>
<td>XP divides its development cycle into; exploration, planning, iterations to release, productionizing, maintenance, and death [15].</td>
<td>Flexibility is one of the core attributes in agile modeling and XP inherits this attribute being a part of agile.</td>
</tr>
<tr>
<td>Adaptive Software Development (ASD)</td>
<td>Development cycle for adaptive software development consists of five activities; initiation phase, adaptive cycle planning, concurrent component engineering, quality review and final quality assurance and release [14].</td>
<td>ASD also inherits flexibility being a part of agile.</td>
</tr>
<tr>
<td>Scrum</td>
<td>Scrum starts from the Pregame phase and then moves to the development phase and finally finishes at the postgame phase [16].</td>
<td>Scrum also inherits flexibility being a part of agile.</td>
</tr>
<tr>
<td>Crystal</td>
<td>Crystal family consists of different methods for different size of projects and these methods have different phases in their project’s lifecycle. Crystal family also supports integration with other agile approaches like XP. In that case, selected agile approach will drive the project’s life cycle [17].</td>
<td>Crystal also inherits flexibility being a part of agile.</td>
</tr>
<tr>
<td>Feature Driven development (FDD)</td>
<td>This approach divides development into four phases namely; develop an overall</td>
<td>FDD also inherits flexibility being a part of agile.</td>
</tr>
</tbody>
</table>
model, build a feature list, plan by feature and design by feature and build by feature. The last phase, i.e. design by feature and build by feature is iterative [18].

| Dynamic Systems Development Method (DSDM) | In DSDM, development is carried out in five phases; feasibility study, business study, functional model iteration, design and build iteration, and implementation. | DSDM also inherits flexibility being a part of agile. |

Table 4.1.2.2 Comparison of Agile Models with respect to Risk Management and Change [23, 24, 15, 17]

<table>
<thead>
<tr>
<th>Models/ Attributes for Comparison</th>
<th>Risk Management</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extreme Programming (XP)</td>
<td>XP manages risk through the phase of productionizing [15].</td>
<td>XP manages change in three phases; iterations to release, productionizing, and maintenance. And accepted changes will update the stories for the next iteration.</td>
</tr>
<tr>
<td>Adaptive Software Development (ASD)</td>
<td>High risk items should be developed as early as possible.</td>
<td>ASD provides concurrent component engineering and quality review phase for managing changes.</td>
</tr>
<tr>
<td>Scrum</td>
<td>In scrum, risks are identified and managed by daily scrum meeting and sprint review meeting.</td>
<td>In Scrum, changes are managed in the development phase and accepted changes will be updated in a product backlog.</td>
</tr>
<tr>
<td>Crystal</td>
<td>Risks are managed using different methods for different size of project.</td>
<td>In Crystal, changes are managed by the iterative phase of the selected method.</td>
</tr>
<tr>
<td>Feature Driven development (FDD)</td>
<td>FDD is more plans oriented as compared to other agile models. And this approach is useful for identifying and managing risks.</td>
<td>FDD manages change in the Design by Feature and Build by Feature phase. Accepted changes are updated in the feature set.</td>
</tr>
<tr>
<td>Dynamic Systems Development Method (DSDM)</td>
<td>DSDM identifies risks in the feasibility phase. Risk analysis and management plan is carried out in the functional model iteration phase.</td>
<td>DSDM supports change at three phases; functional model iteration, design and build iteration, and implementation. Approved changes will be updated in the documents of feasibility and business study.</td>
</tr>
</tbody>
</table>
### Table 4.1.2.3. Comparison of Prescriptive Models with respect to Customer Involvement and Deliverables to the Customer [23, 24, 17]

<table>
<thead>
<tr>
<th>Models/ Attributes for Comparison</th>
<th>Customer Involvement</th>
<th>Deliverables to the customer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Extreme Programming (XP)</strong></td>
<td>Customer involvement is the prerequisite for XP [23].</td>
<td>In each iteration (two-three month), small release of the project is delivered to the customer.</td>
</tr>
<tr>
<td><strong>Adaptive Software Development (ASD)</strong></td>
<td>ASD follows the joint application development approach (JAD). JAD involves customer in the process of design and development.</td>
<td>ASD develops project in components. A component is resulted at each iteration which will be handed to the client for further refinements.</td>
</tr>
<tr>
<td><strong>Scrum</strong></td>
<td>Customer participates in the task related to the product Backlog [23].</td>
<td>Scrum divides project into small sprint and each sprint is delivered to the customer.</td>
</tr>
<tr>
<td><strong>Crystal</strong></td>
<td>Customer is one of policy standards in Crystal. The practices that need to be applied during the development are known as policy standards [17].</td>
<td>In crystal, after every one to four months usable release is delivered to the customer.</td>
</tr>
<tr>
<td><strong>Feature Driven development (FDD)</strong></td>
<td>In FDD, customer participation is low as compared to the other agile approaches.</td>
<td>In FDD, after every 1-2 week executable feature is produced and handed to the client.</td>
</tr>
<tr>
<td><strong>Dynamic Systems Development Method (DSDM)</strong></td>
<td>DSDM supports customer participation in business study phase and design and build phase [17].</td>
<td>DSDM builds project by prototypes and these prototypes are handed to the customer that are eventually evolved into a final project.</td>
</tr>
</tbody>
</table>

### 4.2. EP and Agile

In our literature study, we also found that EP was mostly used as a concept and this concept was applied with a support from other software development models [23, 63]. In the beginning it was applied with iterative/incremental development models and presently it is mostly applied with agile methods [63] as shown in figure 4.1. According to our point of view, EP is more like a concept and it could be achieved with different software methodologies that support this kind of idea for the software development.

Agile is the most talked about and modern software development paradigm. Agile methods are adaptive in nature and iterative development is core of agile methods. In all agile methods, software development is carried out in iterative/incremental fashion and functional/workable piece of code is produced at each iteration as shown in figure 4.1. In EP this functional/workable piece of code is called a prototype as shown in figure 4.1.
In both of the methodologies that is EP and agile, customer plays an important role in all phases of the development as can be seen in figure 4.1. Both these approaches are customer centric. From figure 4.1 one can see that, both these methodologies give more attention to the working piece of code for software product as compared to its documentation. Because of their iterative nature of development, changes are also managed explicitly by both methodologies. Therefore, if a change is requested it could be addressed in the next iteration. Steps involved in EP model are similar to iterative/incremental model as already discussed in chapter 3, section 3.1.2.

4.3. Rationale for the Selection of EP

In software engineering community, prototyping has got acceptance as a reliable model for software creation. It is being listed with more traditional process methodologies, such as Waterfall and Spiral models [34].

For a prototype, it is not necessary to satisfy all the constraints on the final version of the system e.g. a prototype may be expressed in a more flexible language, may be less efficient in both time and space, may run on a machine with more resources, may have limited capacity, and it may not have the same degree of concurrency as the final version [32].

Prototypes can be developed either to be thrown away after having some insight or to evolve into a final product. Each of these approaches has its advantages and disadvantages and the choice depends on the
context of the effort. The throw-away prototype is appropriate in the project acquisition phase where the prototype is used to demonstrate the feasibility of a new concept. The apparent disadvantage of a throw-away approach is the implementation effort on code that will not contribute directly to the final product. There is also an attraction to skip or abbreviate documentation for throw-away code.

The evolutionary approaches generate a series of prototypes resulting in the final software product, it initially contains best known and highly prioritized requirements and then it is changed and augmented as new requirements are discovered. This approach depends on special tools and techniques because it is difficult to produce a prototype without considerable changes to its implementation in order to complete all of the details [32].

EP addresses many issues of market driven and customizable generic products such as requirements overload, requirements ambiguity, loose specifications, too many stakeholders, difficulty in identifying key stakeholders, volatile market, political scenarios etc.

The selected approach helps in gathering correct and consistent requirements. The process provides strength to build quality software by means of the ongoing clarification of existing requirements and the discovery of previously omitted or unknown requirements [34]. According to Sommerville, EP methodology is now the normal technique used for web-site development and ecommerce applications [24, 35]. Prototype-based methodologies are centered on the development of an EP [14, p.174], which ultimately becomes the system [35].

4.4. Conclusions

This chapter explored the different alternatives that are available for the software development. In this chapter we did comparison on these alternatives based on six attributes. This comparison helped us in finding strengths and weaknesses of different software development models and it also helped us to find where we can and where we can not apply a particular software development model. As we know that EP addresses most of the problems for customizable generic product development. Therefore, the main reason for selecting EP is customizable generic products. Furthermore, we are also developing customizable generic product at Tenk AB using EP because it was the requirement from Tenk AB.
5. **Scope Of EP**

Chapter 5 starts by discussing situations where EP suits well in order to give clear picture to software companies to make decision regarding adaptation of EP. Then chapter proceeds by explaining use of EP for customizable generic products. This chapter also discusses advantages and limitations of EP that helps a company to analyze its worst and favorable condition.

5.1. **Situations where EP is Suitable**

There are many situations where EP could be a viable option for the software development. From our literature study and our work at Tenk AB we identified six possible areas where performance of EP is much better as compared to other software development models. In the following subsections we will briefly explain each of these areas.

5.1.1. **Ambiguity and Change in Requirements**

The size of software product is increasing and it is becoming more complex in its nature day by day. This increase in size and complexity of software product leads to unclear requirements and it becomes very difficult for a company to gather all the requirements in a single step. When the requirements are ambiguous and cannot be defined in a single step, then change in requirements may occur in development phase. And these situations are not addressed by most of the traditional software development models.

Since requirements validation becomes very difficult for customer/stakeholders from a written requirements specification document. This problem could be addressed by EP, as defined in chapter 2, EP model builds a software product in iterations and can be started with minimum known requirements. At each iteration of prototyping an executable and functional prototype of known requirements is delivered. Prototypes of the known requirements give the functionality to the stakeholder that enables stakeholders to clearly define what they really need. This gives a built in mechanism to EP for managing requirements change and also handles requirements validation in parallel with development. As EP involves software development in iterations, therefore at each iteration previous requirements become clearer and a new set of requirements could be identified for the next iteration.

5.1.2. **Short Time to Market**

As we know software products can be classified into three categories namely bespoke, market driven (generic), and customizable generic products. The importance of time to market for a software product is different in all of these categories. In market driven and customizable generic products, time to market is more important as compared to bespoke products. In both of these products release time plays very crucial role for the success of the product. In order to capture the market and to be the first in market, beta version of the product with most important functionality can be released. While the user is using the beta version, other functionality can be added and product can be released later. But if a company fails to release a product before its competitor, the company might lose biggest market share. Company can avoid this loss by using EP and giving product in different iterations starting with most important functionality.
5.1.3. Managing Risk of Innovation

At present innovation has become a vital factor for survival of software companies. Failure of the new invented product is the main risk involved in innovation. If innovated product fails to achieve expectations of market then company might face the loss of time, money etc.

By developing new innovated products with EP can reduce this associated risk. Since iterations are the basic feature of EP therefore, the basic idea of the product can be developed in first iteration without too many features. And this first prototype can be tested in market to get market response. If response is positive then more features can be added in product in next iterations. But if the market response is not according to company’s expectations then that idea can be dropped without losing too much money and time involved in the development of full featured product.

5.1.4. Too Many Stakeholders

If too many stakeholders are involved in the software development process then it becomes very difficult to gather requirements from every stakeholder. Involvement of too many stakeholders can also result in more conflicting requirements. Therefore, it is not easy for a company to resolve all of these conflicts. EP solves this problem by giving functional and executable prototypes to stakeholders of the system. Stakeholders find it easy to suggest their updates and comments on working piece of the system. Developing software product in this way will resolves conflicts of many stakeholders.

5.1.5. Software Testing at Early Stages of Software Development

Software testing is the most important activity in SDLC. Research suggests that almost 50 percent of the resources of SDLC are being consumed by software testing [72, 78]. Research has been done to reduce the amount of resources used by software testing [72, 78]. EP could be one of such methods for minimizing resources used in software testing. It does not focus on the development of fully featured system rather the system is evolved from small prototypes. Hence with EP it becomes very easy to perform software testing at early stages of SDLC unlike the traditional software development approaches. In traditional software development approaches, testing is performed at the final stage of software development. If error is detected at this stage then the whole process of SDLC is repeated from initial that consumes a lot of resources.

5.1.6. Testing User Interfaces

Today Graphical User Interface (GUI) is the basic requirement of most of the software systems. User of the system is often more concerned with its GUI. With the system developed using traditional software development approach, it is not possible for user to see its GUI until it is not completely developed. In this case if user does not like user interface of the system and wants it to be changed, then it will require a lot of changes in other parts of the system as well.

This is not the case with EP. With the selected approach user interface can be developed as a prototype and can be used as an agreement between customer and developing organization.
5.2. Advantages and Limitations of EP

In this section we will summarize advantages and limitations of EP from our literature study together with our work at Tenk. This section will be further elaborated in section 7.3 “Evaluating results of EP” by discussing the results of EP after implementing EP on two projects at Tenk.

5.2.1. Advantages of EP

5.2.1.1. Removing Misunderstandings

As literature suggests that prototyping oriented software development involves developing the system in different iterations in the form of functional prototype at each iteration. By Giving system in prototypes allows users to know more about the system before its final delivery. At any functional prototype of the system user can request changes in the proposed system that can reduce misunderstandings between user and development team. During our work at Tenk, we found that by developing a project with EP, provides better communication between the involved stakeholders. Since we achieved the EP by using Scrum, we had daily stand up meetings with other members of team. In this meeting, every team member was to discuss his problems that he/she was facing, what he/she has been doing yesterday and what is he/she going to do today. Daily stand up meetings removed many of the misunderstanding between the team members.

5.2.1.2. Customer Satisfaction

From literature review we come to know that, since user involvement is a key factor in EP oriented development therefore, this approach provides better communication between the development team and end user. Because of this ease of communication user feels closer to software development process. At any iteration user can give comments on some particular functionality. Development team considers these suggestions/comments of the customer/user in the next iteration. Doing development in this way provides great customer satisfaction. We also experienced high customer satisfaction during our work at Tenk. In Tenk, after the completion of each prototype, it was discussed with the customer in order to validate its intended functionality. In this way it removed many of the misunderstandings between customer and developers. This enabled developers to develop right product that meets high customer satisfaction.

5.2.1.3. Problem Identification at Early Stages

From literature we know that in EP oriented development first prototype of the system can be implemented with initial known requirements. This prototype of the initial known requirements helps the end users to explain the developers what they really need and what they really do not need from the given prototype. This explanation of customer’s needs helps the development team in finding problems at early stages of product development. During our work we got the same advantage of EP. In Tenk each project is started with initial known requirements that can be transformed into a final product through a series of prototypes. Developers implemented these initial known requirements into a functional prototype that was later discussed with customer. By discussing this functional prototype (with initial known requirements) helped in identifying problems in the existing requirements at early stages and discovering some more new requirements.
5.2.1.4. Easy Adaptable

We know that to develop a software product/project with traditional software development process requires at least four basic steps in a linear fashion: analysis, design, implementation and testing. With traditional software approaches like waterfall next phase cannot be started until the previous is not finished. This is not the case with EP, where software development process can be started with initial known requirements without much focus on analysis and design. An executable prototype of initial known requirements can be delivered to the customer for testing. After a prototype delivery, some new requirements may be discovered and previous requirements become clearer that will be implemented in next iteration of development. This gives flexibility in software development process. It is very clear from the discussion that the selected approach is useful in situations where the development company does not have a clear idea/scope about the product/project to be developed. Literature also suggests that for the companies that are not following any of the software process models, it is very easy for them to adapt EP model rather the more strict models like waterfall approach. In Tenk this was not the case for the development of our projects. We were quite new in Tenk and did not have enough experience from industry that’s why we found EP difficult to adapt.

5.2.1.5. Better Product Validation

As described in literature review section, in evolutionary software development final system is evolved from small prototypes in different iterations. At each iteration an executable prototype of the system is delivered and given to the user. This prototype is then tested and evaluated by the user. User can request changes to any delivered prototype. From this we can see that in EP, testing is performed at each iteration and hence the final evolved system is highly tested and validated. During our work at Tenk, we developed two projects using EP. In both the projects, customer was involved throughout the development process. After the development of each functional prototype, it was discussed with customer in order to map it with customer’s requirements and willingness. Each subsequent prototype removed a lot of misunderstandings between customer and developers that provided better product validation.

5.2.1.6. Flexibility in Phases

During the literature review, it has been observed that unlike the traditional software approaches like waterfall, EP gives flexibility in switching from one phase to another. In traditional software development approaches all phases are performed sequentially. Starting of next phase depends on completion of previous phase. This is not the case with EP. In EP, software testing is performed at each iteration and new requirements can come at any iteration. This gives flexibility of moving into another software development phase while one is not completely finished. Tenk takes the same advantage of EP by giving the developers the opportunity to focus totally on the development part of the projects. Only little work is done related to design activities. There is no strictness in development phases. Developers can start a project either with documenting some important software issues or by directly implementing the known requirements. Software testing can be performed in parallel with software implementation phase.

5.2.1.7. Focus on Working Piece of Code and Design Flexibility

From literature and our experience at TAT AB we have seen that in EP development process, focus is more towards development of prototype rather than analysis and design. This gives design flexibility during the
development process, especially when there are unclear requirements or when the developers are inexperienced with the system being developed. During the development of our projects at Tenk, we found that at Tenk requires developers to put more focus on working peace of code rather than work related to software design and documentation. Only few design related activities are performed like documenting requirements etc. Our experience show that this kind of design flexibility leads to immature documentation, maintenance problems and poor software quality.

5.2.2. Limitations of EP

Like all other software process models EP also has some limitations. Before selecting any software process model, it is very important to find the limitations of that software process model. Because limitations provide very good analysis how certain software process model would behave in worst conditions. In the following sub-sections, we will discuss limitations of the selected approach. These limitations have been derived from literature review section and from our experience at Tenk.

5.2.2.1. Poor Quality Standards of Code and Maintainability

From literature study we come to know that with EP quality of code for delivered product may be sacrificed. In EP more importance is given to the delivery of prototypes as compared to structure and quality of code for the delivered product. For this reason maintenance and modification of product becomes very difficult for the development organization. Therefore, it becomes very important for the development organization to maintain some kind of quality standards for coding in order to cope with this problem. We experienced same problem during our work at Tenk. After the completion of first project at Tenk, we realized that the quality of code is not good. Concept Lead at Tenk asked us to give the code of Social Location Project to some university students. At that time it took much time to re-arrange the code and to bring it to some position where it could follow some standards. Poor quality of code in the first prototype created a lot of problems in the subsequent prototypes.

5.2.2.2. Requires Experienced and Skilled Manpower

High level design decisions regarding complex programming tasks are often involved in prototyping oriented development. Therefore, it becomes very difficult for inexperienced programmers to make such decisions. This fact is also confirmed from literature study [36] and work at TAT, we found that experience and skill of developer play an important role in the evolutionary prototyping oriented development. Since most of the developers at Tenk are master students like us, they do not have much experience in development from industry. During our first project at Tenk, four developers were involved for its development. Sometimes we all got the situation where we got stuck and we had to explore a lot of material in order to solve that problem. Especially configuration management created lot of problem in the beginning of the project, it was very difficult for us to integrate the changes and updates with each other. But in second project much of these types of problems were reduced due to the experience of previous project. We started to use some special tools to integrate our work with each other.
5.2.2.3. Bad Estimation of Cost and Time

From literature study it is shown that in prototyping oriented development, many delivered outputs are visible at the beginning stage of product development that lead management to think that development of product is just downhill. And this leads to wrong estimation of cost and time for the product by the management. To overcome this problem organization should train their management about the concepts of prototyping oriented development. During our experience at Tenk time and cost estimation for both of our projects went very well. All the deadlines defined by management team were met successfully. When management was asked about the cost estimation, we came to know that the projects were also finished within the allocated budget.

5.2.2.4. Time Consuming

The literature study and our experience at Tenk tell that there are many reasons why prototyping oriented development needs more time than other traditional approaches such as delivery of prototypes in each step of development, quality of code, and customer involvement. Different functional prototypes are delivered in different iterations and this is a core of prototyping oriented development that requires time. Second reason is quality of code that is not maintained in many cases of prototyping oriented development which may produce hindrances in the evolution of prototype as a final product. This also happened with us during our experience at Tenk. If quality of code is not maintained then development time may increase because every evolved prototype is based on successive prototype(s). Third reason is customer involvement, at each iteration a functional prototype is delivered to the customer to get feedback and suggestions from the customer. Next iteration can only be started after having feedback and suggestion from customer. This whole process consumes lot of time and sometimes customer is also busy in other tasks that will add additional delay in the development process.

5.2.2.5. Too Much Customer Involvement

Literature study tells that EP does not give too much focus on the documentation and there is no agreed requirement specification document available that contains all the needs of customer(s). In EP, development does not start with complete requirements rather it can be started with initial known set of requirements. And as the development progresses more requirements are gathered from customer(s). In this kind of development customer(s) can add more requirements than expected and customer(s) drive(s) the development process. This could have negative impact on the development process and it also takes more resources than the expected one and thus reducing the benefit of the development organization.

This was the same case with our experience, In Tenk requirements were not given in documented form for some project or prototype, rather these were just explained by the customer in a meeting. Later on Customer was involved throughout the development process. Sometimes customer added too many requirements at run time creating much problems and disturbance in development process. But overall we observed that customer involvement was proved to be lucrative during our projects.
5.3. Customizable Generic Products and EP

As already mentioned in section 2.4, Software products can be classified in three categories namely bespoke, market driven and customizable generic product. Companies developing market driven or customizable generic products are facing similar type of challenges and problem like requirements overload, requirements ambiguity, loose specifications, too many stakeholders, difficulty in identifying key stakeholders, volatile market, political scenarios etc [1, 3]. Most of the mobile applications come under the category of customizable generic products [79]. Today many companies are developing mobile applications and competition is getting very tough day by day. For the survival of mobile industry it has become necessary for them to be innovative, meet customer requirements, release products in short time periods, reduce cost of development, and design better user interfaces.

From the above paragraph we can see that most of the problems faced by mobile industry are same as situations addressed by EP described in earlier sections 5.1-5.2. Therefore we can say that customizable generic products can be developed in a better way by using EP. This is our finding from literature but this hypothesis is further strengthened by our results from case study (see section 7)

5.4. Conclusions

To conclude this chapter we can say that no single models exists that could fulfill requirements of all software industry. Software development models play an important role in the development of any software product. The selection of an appropriate software development is a crucial factor for the success of software product. These are the reasons why organizations should care about software development models for the development of their software products. Different software models are favorable in different situations. EP can be a suitable choice for developing a software product in situations like ambiguity and change in requirements, short time to market, managing risk of innovation, too many stakeholders, software testing at early stages of software development, testing user interfaces etc. These situations also become its advantages. A company can make better decision regarding selection of EP by having a look on its limitations and its comparison with modern and adaptive software development methodology known as agile.
6. EP and TAT AB

In order to understand and extract information from the results of this case study, it is very important to get background of company (e.g. size of the company, what kind of products are developed by the company, its tools and processes etc). Company environment is discussed because it is very important for us to tell the reader of this thesis about the environment in which case study was conducted. This know-how of company environment will enable reader to decide whether the EP is also good for his/her company if it has same kind of environment.

In this chapter we will explore EP at TAT and Tenk. This chapter starts with the introduction of TAT AB. In section 6.1 we will describe the company, its tools, its history and motivation behind its foundation. This section also describes different products of company and its processes. In section 6.2 we focus the on research and development department of TAT called Tenk. Here we will also discuss Tenk processes and applicability of EP at Tenk. Finally we will discuss our experience and projects at Tenk.

6.1. TAT Introduction

Most of the information here was provided by different people employed at TAT AB. TAT (The Astonishing Tribe) is a Swedish software technology and design company offering products and services in GUI. TAT was founded in 2002 and soon its founders identified mobile industry as the logical target for the collective expertise. At that time, mobile phones were turning from functionality-focused, hardware-centric professional devices into consumer-oriented and designed mass market devices. TAT was founded on a passion for developing digital visual experiences through the combination of aesthetics and technology.

Color screens were new and handset vendors were struggling with the challenge of making the user interface look attractive without adding too advanced, too expensive hardware. TAT’s original core product, Kastor, solved the issue of pushing graphics as fast as possible to screens using the lowest possible processing power and memory. And it solved the problem without making significant changes to the existing software platform. The Astonishing Tribe had found its market and established a foundation on which to build a successful company.

At present TAT is headquartered in Malmo, Sweden, and has local offices in Korea and USA. TAT works with 4 of the 5 leading OEMs in the mobile device space today. Their clients include SonyEricsson, Samsung, TeliaSonera and Orange. TAT has an extensive network of partners covering the whole value chain in the mobile device market. Texas Instruments, Freescale, Teleca, Macnica Networks, Montavista, Nvidia and Symbian are among TAT’s partners.

6.1.1. TAT Products

TAT products are as follows:

6.1.1.1. TAT Cascades™ UI Solutions

User Interface design and integration can be a time consuming and cumbersome process, resulting in negative time-to-market effects and unnecessary cost of resources. TAT Cascades includes a suite of modules – UI References, UI Snippets Library and UI Migration Kit – that can be used to quickly get up to speed in building state-of-the-art UIs.
6.1.1.2. TAT Motion Lab™

TAT Motion Lab is an XML development environment for TAT Cascades. It speeds up the process of crafting rich user interfaces. TAT Motion Lab makes it possible to graphically manipulate visual productions as well as all the powerful editing features for XML. TAT Motion Lab works both as a prototyping environment and as an integrated production environment with TAT Cascades. Combined, they enable designers and engineers to create rich and astonishing multimedia user interfaces on any device, without losing time or quality in the development process.

6.1.1.3. TAT Cascades™

TAT Cascades is a UI framework for the production of advanced user interfaces. TAT Cascades makes it possible to quickly and easily create and customize unique user interfaces with unmatched graphical capabilities, giving consumers a richer and more dynamic experience.

6.1.1.4. TAT Kastor™

TAT Kastor is a powerful UI rendering platform. TAT Kastor is the core UI component and technology. It is a very powerful platform for creating advanced UIs in any type of embedded device environment powering products like TAT Cascades. TAT Kastor sets a new foundation that ensures customer will be able to develop his/her products in an efficient, reliable and flexible environment that grows with customer’s needs.

6.1.2. Processes at TAT AB

![Scrum Process at TAT](image)

Figure 6.1 Scrum Process at TAT

TAT provides its services in user interface mostly for hand held devices. There are almost 150 employees working in TAT. There are four major products of TAT as described above: TAT Kastor™, TAT Cascades™, TAT Motion Lab™, and TAT Cascades™ UI Solutions. TAT is mainly using Atlassian JIRA for requirements/bug handling and planning. Other tools used for this purpose are Mantis, Excel and
Microsoft Project. Microsoft Visio and Photoshop/Illustrator is mainly used when creating images and charts. Implementation (programming) is mainly done in Visual Studio 2005. Some people are using Eclipse, Vim, Emacs. Edit Plus, UltraEdit, Visual Assist X and Beyond compare are also used. Subversion is used for configuration management. Designers at TAT are using mainly Adobe Photoshop and AfterEffects for GUI design purposes. For testing, Atlassian Bamboo is used to control builds and continuous integration. A number of in-house tools are also used. TAT also uses Confluence as the base for both the external developer site (product documentation, FAQ, soon deliveries, etc) and the new intranet.

TAT uses Scrum as a process model for its product development. Fig 6.1 shows the Scrum process at TAT. We will here provide some details of Scrum at TAT AB, because we will also use this approach for the development of our own prototypes for our case study. Details of Scrum at TAT are as follows:

- There are two types of actors involved in scrum during development process, one is scrum master and other one is developer who is part of self organized team known as scrum team.
- Scrum master is responsible for product backlog and sprint planning. A product backlog consists of different features and services of a product to be developed. Scrum master is also responsible to make sure that all the process is followed in right way. During all the development process he acts as a firewall for the scrum team. Scrum master has to take quick decisions (almost within an hour) regarding sprint tasks. Fig 6.2 shows how Scrum master at TAT manages his responsibilities.

![Figure 6.2 Management view of Scrum at TAT](image)

- A scrum is divided into different sprints and mostly each sprint is of two weeks. For each sprint one or more items are taken from product backlog depending on estimated development of item so that sprint could be finished within two weeks.
- Each selected item(s) from product backlog is further divided into different subtasks that form sprint backlog. To start sprint, items from sprint backlog are selected by the self organized scrum team members depending on their choice.
• Daily stand up scrum meetings are conducted for a very short time (almost 10 to 15 mints). In the scrum meeting every team member has to answer following questions:
  - What have you done since last day?
  - What will you do today?
  - What is blocking you?
  - Any new task to sprint backlog?
  - Have you learnt or decided anything new?

• A working product is produced at the end of each sprint.
• A sprint review is conducted to analyze what scrum team accomplished during the sprint and outputs from sprint are also assessed against selected sprint goals. Actors involved in this review are product owner, scrum team and scrum master.
• In sprint review total assigned work effort is compared with actual total taken team effort with help of Burndown chart as shown in Fig 6.3. In the chart, along y-axis time in remaining hours is represented while along x-axis total allocated days are presented. In the chart red line shows allocated 140 hour of work in twenty days. Green line shows the team effort in hours per day and blue line shows completed work in hours in a day. Yellow line in chart shows the estimate accuracy for the allocated time against actual taken time.

![Burndown chart](image)

Figure 6.3 Burndown Chart Used in TAT
6.2. Tenk AB

TAT has a separate department for research and development called Tenk. Tenk almost consists of 15 peoples working in different areas. In Tenk most of the work force is thesis students from different universities working in different research areas. Only three out of fifteen are permanent employees at Tenk. The main motivation behind the establishment of Tenk was to prototype an idea generated by TAT or Tenk itself. Tenk is also looking for different future openings for TAT. All our research work and projects were accomplished at Tenk.

6.2.1. Processes at Tenk AB

Tenk AB was recently setup as a research and development department of TAT AB. The main motivation behind Tenk AB was to prototype an idea that could become a future product of TAT AB. Tenk AB does not have very well defined processes as it is still very new and immature. EP was selected as an approach for the development of projects at Tenk AB. Tenk AB was using EP as a strategy not as a complete process model as defined by Boar and Bischofberger in [64, 70]. Sommerville [24] and Larman [63] have also suggested that EP is not used alone as a complete process model rather it is supported with an iterative approach. In Tenk AB, EP was supplemented with Scrum methodology as mentioned by the Sommerville and Larman. The reason to support EP with Scrum are similarities between agile and EP, these similarities include iterative nature of development and adaptive to the changing requirements as mentioned in [60, 61, 62].

In Tenk AB, as we observed that developed prototype contains 40% of the functionality of the original product. And if, management finds it feasible then they start implementing the rest of functionality in that prototype, to transform it into a complete product. Other processes of Tenk AB were also not very well defined, no requirements document exists and requirements were given to the developer during the meeting. All documentation related to design and features are maintained in the Google docs. Testing was also carried out by the developers and its results are also maintained in the Google docs. But testing was carried out in parallel to the development as mentioned in [69].

At Tenk, ideas are prototyped and then analyzed based on four attributes: need, approach, benefit versus cost, and competition. Tenk calls this analysis as NABC. The decision regarding evolution of prototype into a final product is based on NABC analysis results. In Tenk requirements are not given in a written document rather the basic idea for a prototype is explained to a thesis worker. Further clarification of requirements regarding that idea is achieved via the workable prototype and face to face meetings. Tenk uses Scrum model for prototypes development. Tenk also holds a weekly meeting called “Monday meeting” in every week. All the Tenk workers attend this meeting and everyone shares what he/she has been doing last week and what he/she has plans to do in next week. The main purpose of this meeting is to share what is the current status of each project at Tenk.

A brain storm meeting can also be arranged on the request by any of the project owners when they feel stuck or need some ideas or help. Thesis workers also have informal discussions among them. At present, Tenk is using JAVA and TAT Cascades™ for the development of prototypes, confluence and Google docs for documentation sharing, and SVN for configuration management.
6.3. Applicability of EP at Tenk

The main aim of our thesis is to perform an initial validation of EP for customizable generic products in industry. Therefore, we will discuss the applicability of the selected approach in our projects at Tenk. As described in chapter 5, section 5.4, evolutionary prototyping is a concept that can be implemented using any iterative methodology. Tenk accomplishes concept of EP by using Scrum methodology of Agile as a process model. Tenk is different from TAT in a sense that it does not develop complete products rather it shapes an idea into a functional prototype.

6.3.1. EP and Our Projects at Tenk

In Tenk, the main theme behind prototypes creation is to build an idea or to clear an idea and its requirements for later stages of development as shown in figure 6.4. We implemented our projects with the concept of EP using Scrum as development model for the development of prototypes. The motivation behind selection of Scrum was the workshop conducted by senior scrum master of TAT. This workshop was about the Scrum model and its practices at TAT. The Scrum process that we were following in Tenk was almost the same as being followed in TAT.

The only difference that we observed in the Scrum process of TAT and Tenk was product backlog. During our research and experience at Tenk, we found that when scrum is applied for a product with more clear requirements and functionality then the product backlog list becomes very large as was the case with TAT. While if the same model is applied for developing a prototype for vague idea with unclear requirements then the product backlog list is too small to exist that was the case in Tenk. In this case scrum master have only sprint backlog not the product backlog. This was our experience with Social Location Project as shown in figures 6.4 and 6.5.

![Scrum Process at Tenk](image)

Figure 6.4 Scrum Process at Tenk

In case of product backlog, the source for sprint backlog is one or more item(s) from product backlog, but as described above in case of no product backlog, the source for sprint backlog is an idea (clear or vague). Completion of each sprint may give birth to new ideas or make clearer the existing idea as shown in figure
6.4. In first case, number of sprints is known because of available product backlog and one knows when to stop development process. In second case with no product backlog, one cannot estimate total number of sprints and does not know when to stop development process.

As soon as an idea is converted into a sprint backlog, all the items of sprint backlog are pasted as sticky notes by Scrum master on the whiteboard in his room as shown in figure 6.5. As described earlier that Scrum team is self organized therefore, all members of Scrum team can select different tasks based on their choice as shown in figure 6.6.
During our experience at Tenk, we developed two projects namely Social Location and Social Playlist. In Social Playlist two other developers were also involved with us in development. But Social Location project was solely developed by us. Idea of the Social Playlist project was very much clear at initial stages and final product was evolved in three iterations. Still work is going on this project and further iterations may be expected in future. On the other hand in Social Location project, even the idea was very vague. After the completion of second iteration of Social Location project, different ideas for next iterations were emerged. For us, it became very difficult to decide one idea that could be implemented in next iteration. To solve this problem brain storm meeting was requested with all thesis workers and with concept lead of Tenk. During the brain storm meeting four ideas were finalized. To decide one of these four ideas another meeting with Co-Founder of TAT (who was also our supervisor in this project) was arranged. After this meeting we decided one idea to implement as a next iteration of the project. In this way we completed our thesis task in three iterations. But Tenk still wants further additions in that project and may be next iterations of this project will be continued by some other thesis workers. During our work at Tenk AB, we have found that EP is paying off because it helps in early acceptance of mobile applications. This fact was also confirmed by Schwaiger [44].

6.4. Conclusions

TAT AB was founded in 2002 with the motivation of better looking GUIs in hand held devices with limited memory and processing capabilities. Main products of TAT are: TAT Cascades™ UI Solutions, TAT Motion Lab™, TAT Cascades™, and TAT Kastor™. TAT is using Scrum as process model for its product development. Innovation and rapid change in customer needs and market inspired TAT to follow Scrum that handles these problems better than other software development approaches. Continues innovation has become a key to survival for mobile companies. This forced TAT to have its separate department that could manage future innovations known as Tenk. Tenk has lot of influence from TAT, therefore it is also using Scrum for its prototype development. In Tenk, an idea for innovation is evolved into a product through different prototypes that is similar to EP development. At Tenk, we achieved the concept of EP with the help of Scrum.
7. **INITIAL VALIDATION OF EP FOR CUSTOMIZABLE GENERIC PRODUCTS**

During our work at Tenk we have developed two applications using EP. Here we will present results that we got by developing applications through EP. Results will be presented in first section of this chapter. In first section we will discuss that how the developed products are customizable generic products. Based on data from the second section, in the third section, evaluation of EP will be performed against different attributes of software process models. In section 7.4 we will draw conclusions from the results based on data and our observations during our work at Tenk. In section 7.5 we will discuss some known threats to our study. Finally we will conclude this chapter in section 7.6.

7.1. **Social Playlist and Social Location as Customizable generic products**

Both the developed projects come under the category of customizable generic products. These products are developed by using loosely coupled components, so that any new features could be easily added in the existing product. For example, Social location application facilitates a user to know about his/her current social location. This project has also the flexibility of adding some features so that it could be used to monitor office timings of employees of a company. Client side of this application remains similar in both the cases, while the server side is changed according to the requirements. This was just an example that why the developed applications come under customizable generic products. There could be many features like this in both the applications that make them customizable generic products. Now it depends on customer what features he/she wants to be implemented.

7.2. **Research Settings (Conditions, Environment, and Data) of EP**

As mentioned in section 5 of first chapter, we have selected action research for our study because we were ourselves involved in the development of both of these applications Action research combines researchers and practitioners to act together on a particular cycle of activities including problem diagnosis, action intervention and reflective learning [84]. Action research is more focused on observable effects as compared to the structural and data analysis [84]. The same position is taken in our research and therefore most of the results presented in this thesis are from observation. The data was collected by observing people how they perform certain tasks rather than by asking them. As we were focused on how practitioners do certain tasks not on what they say, that is the reason for selecting action research as a research methodology for our study. In this perspective, we can say that our research is different from other’s research. Lot of other research is focused on asking practitioners why they do certain task differently (via interviews, questionnaire etc). The results presented in this thesis are not objective rather these are subjective [83].

In this section, we will present data that we got after the development of two applications using EP. It is also very important to explain the environment and conditions under which development was carried out. In the first part of this section we will present conditions and environment for the development. Then in the second part we will present data of EP from our developed applications.
The gathered data is organized and analyzed based on the approach used by Abrahamsson [74] for evaluating XP in real environment. The analysis of results is also carried out using the approach defined in [74].

### 7.2.1. Conditions and Environment of Development

In order to achieve our objective of research we have developed two different applications namely Social Playlist and Social Location at Tenk. We were partially involved in the development of Social Playlist. Social Location was completely developed by us.

#### 7.2.1.1. Conditions and Environment of Social Playlist

Social Playlist is a prototype of an idea built in three iterations with an estimated effort of 330 man hours. Social playlist can be a future customizable generic product of TAT. Social Playlist was developed by team of six members of Tenk, four developers and two designers using J2ME, Servlets, and JSP. Tenk was the only stakeholder for Social Playlist because it was based on innovative idea with no intended customer. Idea owner of Social Playlist was the main source of requirements for Social playlist. For this project requirements were not given in a written document. Rather requirements for each iteration were briefed in meetings. During the development process, if a requirement was not clear to developer, then it was made clear by face to face meeting between developer and the requirement source. After first iteration, SVN was used as a configuration management system for Social Playlist.

Tables 7.1 and 7.2 present qualification and experiences of the team members involved during the development of social playlist project.

<table>
<thead>
<tr>
<th>Member</th>
<th>Qualification</th>
<th>Experience</th>
<th>Scrum Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer1</td>
<td>BS in Computer Science, and doing MSc in Software Engineering</td>
<td>Six month experience as an internee in a software house</td>
<td>Nil</td>
</tr>
<tr>
<td>Programmer2</td>
<td>BS in Computer Science, and doing MSc in Software Engineering</td>
<td>One year experience as an application developer in a software house.</td>
<td>Nil</td>
</tr>
<tr>
<td>Programmer3</td>
<td>MSc Computer Science specially in softwares</td>
<td>One and half year experience of development in industry, XP coach during his study</td>
<td>NIL</td>
</tr>
<tr>
<td>Member</td>
<td>Qualification</td>
<td>Experience</td>
<td>Scrum Experience</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Designer1</td>
<td>Master in Human Computer Interaction, Doing Masters in Interaction Design</td>
<td>One and half year experience as Researcher and Designers in Antivirus software company, one and half year experience of user experience research at YAHOO.</td>
<td>Nil</td>
</tr>
<tr>
<td>Designer2</td>
<td>MSc in Interaction Design</td>
<td>Three years experience as web designer, 6 month experience as interaction designer</td>
<td>Nil</td>
</tr>
</tbody>
</table>

7.2.1.2. **Conditions and Environment of Social Location**
Social Location is also a prototype of an idea built in three iterations with an estimated effort of 330 man hours. It was built using J2ME, Servlets, and JSP. This prototype has potential to become future customizable generic product of TAT. Social Location was developed by team of four members of Tenk out of which two were developers and two were interaction designers. Programmers of this application also got programming support from other Tenk members. Also in Social Location, Tenk was the only stakeholder of project because the application was based on innovative idea with no intended customer. Idea owner of Social Location was the main source of requirements for Social playlist. For this project requirements were not given in a written document. Rather requirements for each iteration were briefed in meetings. During the development process, if a requirement was not clear to developer, then it was made clear by arranging face to face meeting between developer and the requirement source. SVN was used as a configuration management system for Social Location from the very first iteration.
Tables 7.3 and 7.4 present qualification and experiences of the team members involved in the development of social location project.

Table 7.3 Developers’ Skills and Experiences

<table>
<thead>
<tr>
<th>Member</th>
<th>Qualification</th>
<th>Experience</th>
<th>Scrum Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmer1</td>
<td>BS in Computer Science, and doing MSc in Software</td>
<td>Six month experience as an internee in a software house</td>
<td>One and half month experience</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programmer2</td>
<td>BS in Computer Science, and doing MSc in Software</td>
<td>One year experience as an application developer in a software house.</td>
<td>One and half month experience</td>
</tr>
<tr>
<td></td>
<td>Engineering</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.4 Interaction Designers’ Skills and Experiences

<table>
<thead>
<tr>
<th>Member</th>
<th>Qualification</th>
<th>Experience</th>
<th>Scrum Experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designer1</td>
<td>Master in Human Computer Interaction, Doing Masters in Interaction Design</td>
<td>One and half year experience as Researcher and Designers in Antivirus software company, one and half year experience of user experience research at YAHOO.</td>
<td>Nil</td>
</tr>
<tr>
<td>Designer2</td>
<td>Doing MSc in Interaction Design</td>
<td></td>
<td>Nil</td>
</tr>
</tbody>
</table>
7.2.2. *Data Collected From Social Playlist and Social Location Projects*

Table: 7.5 Attribute values of software process model collected from social playlist projects

<table>
<thead>
<tr>
<th>Data Elements</th>
<th>Prototype 1</th>
<th>Prototype 2</th>
<th>Prototype 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint Backlog (item)</td>
<td>6</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>Allocated time (hour)</td>
<td>115</td>
<td>110</td>
<td>160</td>
</tr>
<tr>
<td>Actual time spent (hour)</td>
<td>105</td>
<td>95</td>
<td>130</td>
</tr>
<tr>
<td>Total lines of code (LOC)</td>
<td>1018</td>
<td>1456</td>
<td>2162</td>
</tr>
<tr>
<td>Team Productivity (LOC/hour)</td>
<td>9.69</td>
<td>15.33</td>
<td>16.63</td>
</tr>
<tr>
<td>Item Effort (mean)</td>
<td>17.5</td>
<td>9.5</td>
<td>7.2</td>
</tr>
<tr>
<td>Item Effort (max)</td>
<td>29</td>
<td>31</td>
<td>19.5</td>
</tr>
<tr>
<td>Average time spent on a task (hour)</td>
<td>17.5</td>
<td>9.5</td>
<td>7.22</td>
</tr>
<tr>
<td>Max effort spent on a task (hour)</td>
<td>25</td>
<td>16</td>
<td>14</td>
</tr>
<tr>
<td>Actual time spent /Allocated time *100</td>
<td>91%</td>
<td>86%</td>
<td>81%</td>
</tr>
<tr>
<td># post-release defects (item)</td>
<td>5</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Post-release defects/K-LOC</td>
<td>4.9</td>
<td>6.2</td>
<td>3.7</td>
</tr>
</tbody>
</table>

Table: 7.6 Attribute values of software process models collected from social location project

<table>
<thead>
<tr>
<th>Data Elements</th>
<th>Prototype 1</th>
<th>Prototype 2</th>
<th>Prototype 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint Backlog (item)</td>
<td>5</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>Allocated time (hour)</td>
<td>90</td>
<td>105</td>
<td>140</td>
</tr>
<tr>
<td>Actual time spent (hour)</td>
<td>90</td>
<td>102</td>
<td>128</td>
</tr>
<tr>
<td>Total lines of code (LOC)</td>
<td>937</td>
<td>1363</td>
<td>1889</td>
</tr>
<tr>
<td>Team Productivity (LOC/hour)</td>
<td>10.41</td>
<td>13.36</td>
<td>14.76</td>
</tr>
<tr>
<td>Item Effort (mean)</td>
<td>18</td>
<td>12.7</td>
<td>9.8</td>
</tr>
<tr>
<td>Item Effort (max)</td>
<td>27</td>
<td>32</td>
<td>18</td>
</tr>
<tr>
<td>Average time spent on a task (hour)</td>
<td>18</td>
<td>12.75</td>
<td>9.84615385</td>
</tr>
<tr>
<td>Max effort spent on a task (hour)</td>
<td>23.5</td>
<td>19</td>
<td>21</td>
</tr>
<tr>
<td>Actual time spent /Allocated time *100</td>
<td>100%</td>
<td>97%</td>
<td>91.5%</td>
</tr>
<tr>
<td># post-release defects (item)</td>
<td>7</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>Post-release defects/K-LOC</td>
<td>7.4</td>
<td>8.8</td>
<td>4.8</td>
</tr>
</tbody>
</table>
7.3. Evaluating the Results of EP

A software process model can be evaluated by different set of attributes. We selected some attributes from [74, 75, 77] and some from our own findings to evaluate EP. Here we will make an evaluation of EP based on time estimation, documentation, software testing, quality of code, quality of product, customer involvement and satisfaction, team effort, delivery time, and change management and overload.

7.3.1. Time Estimation

Time estimation is an important attribute for the measurement of any software process model [74, 75, 77]. In Tenk, there are two types of time estimation: management time estimation and development time estimation. Management time estimation estimates time for management planning, formal meetings, user meetings, brainstorming, defining requirements, user testing, and other activities like these. Development time estimation estimates time required to develop a functional prototype and its pre-release test.

7.3.1.1. Time Estimation for Social Playlist and Social Location projects

- **Management Time**
  
  For the development of both the projects, management time constituted 30 percent of the total allocated time for the projects. As reported by management team, their most of time was spent for defining requirements, formal meetings with development team, and user testing. When we spoke to management team about their time estimation for the projects, they told that time spent on management activities was less as compared to the total allocated time for management activities. For the first iteration management team spent 25 percent time for Social Playlist project and 30 percent for Social Location project of the allocated 30 percent time. For the second iteration management spent 20 percent on social playlist and 24 percent on social location project. In third iterations these times were reduced to 17 percent for Social Playlist and 21 percent for Social Location. From the results of both the projects it can be seen that time estimation for management activities went very well by using EP.

- **Development time**
  
  As it is clear from table 7.5 and table 7.6 that time spent by the development team for the development and pre-release testing for both the projects was less than the total allocated time. Furthermore, from the tables 7.5 and 7.6 it can be seen that for first prototype 91 percent time for social playlist and 100 percent time for social location was spent of the total allocated time for the development and pre-release testing. For second and third prototypes development was reduced to 86 for Social Playlist and 97 percent for Social Location and 81 percent for social playlist and 91.5 percent for social location respectively. Results from both the projects show that by developing a project with EP enables development team to meet their deadlines. From the results, we can also see that the development time reduces with subsequent releases (prototypes).

7.3.2. Documentation

Good and detailed documentation is very important for quality of software product. Documentation removes misunderstandings, conflicts, and ambiguity by defining clear set of requirements, software
product design, and its architecture. In addition to this documentation is also very helpful for the maintenance of software product.

7.3.2.1. Documentation of Social Playlist and Social Location Projects:

In Prototyping oriented software development, more focus is given to the working piece of software as compared to documentation. This was the same in our case for development of social playlist and social location projects. Requirements were not given in a written document except the first prototype of social playlist; rather an idea for the prototype was explained in meetings. For the first prototype of social playlist requirements were given in a very informal way written in Google doc. Developers extract requirements from the explained idea and noted it down on sticky notes to form a sprint backlog. Scrum team members then pick different tasks from sprint backlog to start development process. After the completion of sprint, sticky notes were not kept in any kind of record rather these were wasted. Design document was created only in the first prototype of social playlist in the form of screen layout to explain GUI to developers but it was missing in later prototypes of both the projects. Same was the case with software testing documentation, Google docs were used to note down the reported bugs in different prototypes.

From the aforementioned facts from our projects one can know that documentation is very weak by developing software product with EP.

7.3.3. Software Testing

Software testing is key process for the validation of software product and ensures the quality of software product. It is a known fact that software testing consumes almost 50 percent of the development resources [72, 78].

7.3.3.1. Testing in Social Playlist and Social Location Projects

For both the projects, three types of software testing were used, functional testing, pre-release testing and user testing. Since customer was involved in the development process of both the projects therefore, functional testing was performed in parallel with development phase. Pre-release testing was performed after the completion of each prototype in order to make the prototype more stable and less vulnerable. User testing was performed by the management team in order to find the worth of prototyped idea.

With EP, functional testing is performed at early stages with customer involvement. This testing at early stages of development causes to discover bugs/problems at early stages of development that will reduce the resources and cost of software testing. The results from the user testing were used to make an analysis of idea either to continue with development of idea or drop it. And by doing so company can test different ideas for their innovation without spending cost and resources needed for the development of complete product.

7.3.4. Quality of Code

Quality of code affects software maintainability, software testing and software quality itself. Good code can give reusability of some software components, readability and understandability.

7.3.4.1. Quality of Code in Social Playlist and Social Location Projects
In both the projects quality of code was not good. This was because the focus was more towards
development of projects rather to follow some standards and documentation. Another factor for weak
quality of code was less experience of developer in development field. The quality of code might be good if
the developers would have more experience of development.

From the results of our projects at Tenk, we can see that software product developed using EP does not
necessarily yield good quality of code.

7.3.5. Quality of Product

There is no single/direct factor to measure quality of software product. Software quality depends on many
factors such as documentation, quality of code, customer satisfaction, testing etc. At present, it is very
important for software companies to produce quality in their products due to increase in competition.

7.3.5.1. Quality of Product in Social Playlist and Social Location Projects

In Social Playlist project, quality of the end product was average. But the quality of Social Location project
was better as compared to Social Playlist and this was because of some experience from development of
Social Playlist project. Customers were well satisfied with intended functionality of both the products. This
was because of their involvement throughout the development process of both the projects. Software testing
was also performed very well due to high availability of customers during development process. During the
development process of our projects, customers could suggest any improvement/suggestion at any time.
Customers participated in all kind of testing mentioned earlier. This gave high satisfaction to customer at
the end of these projects. As mentioned earlier that documentation and the quality of code were not good,
these affected the overall quality of products negatively.

7.3.6. Customer Involvement and Satisfaction

Most of the software industry says that a software product is said to be successful if it satisfies its
customer’s needs. Customer satisfaction can be achieved by different ways like delivering product with
intended functionality of customer, product delivery in time, providing cost effective solution etc.

Since in EP customer is involved throughout the development process, this allows customer to see
functionality of the prototype before its delivery. Knowing the functionality before its delivery improves
customer’s understanding about the problem that helps customer in knowing what is possible to implement
and what is not.

7.3.6.1. Customer Involvement and Satisfaction in Social Playlist and Social Location Projects

Since we used EP for the development of Social Playlist and Social Location projects, the customers were
involved at all phases of development cycle. During the development process there were some cases of
Social Playlist project in which some requirements were difficult to implement within time span. But as
customer himself/herself was involved throughout the development process of social playlist project that
removed a lot of misunderstanding between customer and developer. Customer involvement also enabled
them to request changes in the proposed idea. Development team always considered customer’s opinion
during the development of project. Furthermore, this involvement of customer with development team
helped them to know the current status of the project. All these factors increased the level of customer satisfaction.

7.3.7. Team Effort

Team effort can be defined by number of different perspectives. One such perspective can be total number of hours spent on a task by all the members of the team. Another perspective could be number of lines of code produced by the team in an hour. Another way for finding team effort is by comparing ratios between actual spent times with allocated times for different subsequent prototypes.

7.3.7.1. Team Effort for Social Playlist and Social Location Projects

From tables 7.5 and 7.6 we can see that in both the projects total number of hours spent on a task by the team is decreasing for later prototypes. Tables 7.5 and 7.6 also show that the number of lines of code produced per hour by the team is increasing for successive prototypes in both the projects. The calculated ratios between actual taken times with allocated times in all different prototypes of both the projects show regular decrease in team effort as shown in tables 7.5 and 7.6. All these three factors show the positive trend towards the team effort for EP.

7.3.8. Delivery Time

In different scenarios, delivery time is used differently. Here delivery time means the specified time for the delivery of a prototype. Delivery time helps in getting customer confidence and increasing market worth. Specifically, in market driven and customizable generic products the delivery time is very crucial factor for the success of a product.

7.3.8.1. Delivery Time of Social Playlist and Social Location Projects

From the tables 7.5 and 7.6 we can see that for every prototype of Social Playlist and Social Location projects, the actual taken time was less than total allocated time. Therefore we can say that by developing a product through EP ensures in-time delivery of a product.

7.3.9. Change management and Overload

In software development change in requirements is always inevitable. Every software development model has its own way to manage change. Some process models have very complex process to manage change and even resist the change. EP approach implements requested changes in later prototypes of product that gives flexibility in change management. But sometimes too much customer involvement and too many change requests from customer cause an overload to the development team.

7.3.9.1. Change Management and Overload in Social Playlist and Social Location Projects

In social playlist social location projects changes were managed by maintaining a “to do” list document that was shared on Google docs with all the team members. Proposed changes were written in the “to do” document during the development of a prototype. A meeting was conducted between the developers and customer to make an analysis on all proposed changes. After the analysis changes to be implemented were finalized for the next iteration. This whole process adds flexibility in change management. As social
playlist and social location were small projects, therefore too many changes and new requirements were not emerged during the development. So in our case overload did not occur. But in large projects it may occur. The overall

![Effort Distribution](image)

**Figure 7.1 Effort Distribution for Social Playlist and Social Location Projects**

### 7.3.10. Effort Distribution for Social Playlist and Social Location Projects

Effort Distribution for both the Social Playlist and Social Location Projects can be seen in figure 7.1. In both the projects Effort is divided into sub activities performed in order to complete these projects such as: Coding J2ME, Coding Web, Project meetings and planning, Testing, Pre-release tuning, Project management, Refactoring and Configuration management. As shown in figure that coding in terms of testing, pre-release tuning, production code and refactoring took 72.6% of the total effort. Project Planning and meetings took 16.6% of the total effort and project management including configuration management took 9% of the total effort. From the calculations, we can see that coding of both the projects took more time of the total effort than all the other activities as also discussed in [23,63]. There were no formal written documents for the project but architectural description and sprint items were noted down on white board in developer’s room. Architectures designed on white board were stored as picture format for later references.

### 7.4. Results Analysis

In this section we will analyze EP based on the results that we got after developing our projects using EP at Tenk. We will highlight the situations with both the good and bad results of EP.

#### 7.4.1. Innovative Ideas
From experience of Social Location project we found that EP is a very good approach for developing an innovative product. We came to this conclusion because at start of project even the idea was very vague. But after the development of first prototype, a lot of other dimensions regarding the idea were explored that helped in the development of later prototypes of the project. Another finding was the resources used for development of Social Location and Social Playlist. In both the projects, utilization of resources was very less as compared to a utilization of resource in the development of a full product. This fact suggests that the risk of product failure involved in the development of an innovative idea is mitigated with the help of EP.

7.4.2. Ambiguous and Sketchy Requirements

As described earlier that in social location project requirements were unclear and not completely known at first iteration. But after the first functional prototype, requirements became more clear and concrete. We also found that working piece of code (prototype) helps in making concepts and ideas clearer because one feels easier to understand and work with functional entity. Thus we can come to the conclusion that developing a product using EP makes requirements more clear and concrete.

7.4.3. Discovery of New Requirements

It is already described that both the projects were based on innovative ideas, especially Social Location project. At the beginning of Social Location project very few requirements were known. But as the development progressed to later prototypes, a lot of new requirements were discovered. In social Playlist project most of the requirements were known at initial stage. But the release of each prototype still introduced some new requirements for next prototypes. From the mentioned experience of both projects we can see that EP can accommodate discovery of new requirements.

7.4.4. Customer Involvement and Satisfaction

As described by literature, product developed using EP has high customer satisfaction. This was the same with our experience. We found that due to involvement of customer throughout the development process provided high level of satisfaction at the end of product. Form this we can conclude that EP provides great customer satisfaction.

7.4.5. Software Testing

We performed three types of testing during our work at Tenk. Customer was involved in all types of testing. Customer was involved throughout the development process therefore we performed functional testing in parallel with development process. Performing testing in this way helped us a lot to find problems/bugs at early stages of the software development. This early detection of problem enabled us to remove bugs at early stages which saved time and resources. User testing identified risk associated with the idea. Development process can be stopped at any iteration (prototype) if there is some risk involved in final product. Dropping an idea at any iteration will save resources, time, and cost associated with the complete development of product. Hence from our discussion we suggest the following conclusion. Software testing with EP does not consume too much resource, cost, and time as compared to other software process models.
7.4.6. *Software Quality*

We already know that there is not a single factor that determines the quality of software product. There are many factors that can be used to measure the quality of software product such as good documentation, quality of code, customer satisfaction, testing etc. From our experience, the quality of both the projects was average because some of these factors were affecting quality positively while some were affecting negatively. For example as described above customer satisfaction was very high while the documentation was very weak.

7.5. **Threats to Study**

A threat in a research can be defined as how valid results are from that research [76]. The know-how of these threats for the study is very important in order to mitigate these threats in the research. In our study we identified two types of threat that is general validity threat and threat in findings.

7.5.1. **General Validity Threat**

We are aware that results from case studies cannot be generalized because it explores a case in a single situation. This kind of threat also exists in our case, since we also conducted our case study in one company. We tried to reduce this threat by conducting our case study for two different projects at TAT AB. Since the domain of both the projects is different therefore, we can say that the results of this case study are reliable to some extent. We also tried to lessen this threat by discussing environment, process and conditions under which results were gathered and analyzed (e.g. see sections 6.2, 6.3 and 7.1).

7.5.2. **Threat in Findings**

Researchers were themselves involved in the field work. Therefore, there could be a possibility of threat in the findings of this case study. The opinion of the researchers can be biased due to their involvement in the field work and this may affect the findings of the study. We were aware of this kind of threat and we tried to mitigate this threat by discussing collected data with other members of Tenk that were not involved in this research. Final finding that are presented in this case study were discussed these with other members of Tenk. This helped to reduce the threat to some-how.

7.6. **Conclusions**

From our literature study we found that EP addresses many problems of market driven products. We also found many case studies validating EP for market driven products. Customizable generic products are facing similar kind of problems as market driven products. We made an initial validation of customizable generic products in this chapter. From the conclusions of the presented results and our observations during our work at Tenk we can conclude that EP addresses many of the problems of customizable generic products. From our findings it shows positive trends towards innovation, ambiguous and sketchy requirements, discovery of new requirements, software testing and customer involvement and satisfaction. But there were some factors that show the negative trend of the selected approach like documentation, quality of code.
8. **CONCLUSIONS AND FUTURE WORK**

This chapter constitutes a summary of our work, together with conclusions and discussions on our findings of the research that we conducted at Tenk. We will also discuss possible future work that can be done in this field.

8.1. **Answers to Research Questions**

In this section we will briefly summarize all the research questions identified in the beginning of our study.

**RQ1:** What literature offers about prototyping and EP?

Answer: From the study we have come to know that prototyping is a technique for software development in which emphasis is given on developing prototypes early in the development process to get early feedback and analysis to carry development process. EP is process in which a software product is developed through a series of prototypes. A lot of literature has been discussed on EP; a brief summary of EP has been given in chapter 3 explaining prototyping and EP practices from 1990s till now. Chapter 5 explains the scope of EP by discussing its advantages and disadvantages and exploring the situations where EP is a good choice.

**RQ2:** What other alternatives are there of EP for software development?

Answer: There are different alternatives of EP available for software development like Waterfall Model, Incremental Process Models, Specialized process Models, The Unified Process etc. Each process model has its own advantages and disadvantages over EP. In chapter 4 we have identified different alternatives of EP for software development and have made a comparison in order to know the situation where a particular model fits.

**RQ3:** How EP is used by Tenk AB?

Answer: As the main aim of this study was to evaluate EP for customizable generic products in Tenk AB, and the research was conducted in Tenk (R&D of TAT AB) by developing two customizable generic products. Therefore it was of utmost importance to explain all the processes, methods and environment of the company itself in order to understand the results of the study in more beneficial way. Tenk is using EP for the development of its software products. Tenk implements EP using Scrum methodology of Agile Technology. Chapter 6 gives a brief introduction of the company, its products, processes and tools used to develop its products. The chapter focuses on processes of Tenk and applicability of EP at Tenk.

**RQ4:** What are strengths and weaknesses of EP for different types of products (bespoke, market driven and customizable generic products)?

Answer: EP has many advantages for developing market driven and customizable generic products. As mentioned in Section 7.4 and 4.3 that EP is very good approach for the development of software products that involve innovative ideas, ambiguous requirements, loose specification, too many stake holders, volatile market and incomplete set of requirements at initial stages. Since both market driven and customizable generic products are facing similar kind of problems, Therefore we can say that EP is a very good choice for developing these kinds of products. Section 4.3, 5.2 , 5.3 and 7.4 of the study explains that how the EP is good for customizable generic products.
RQ5: Is it useful for Tenk AB to build customizable generic products using EP?

Answer: As we have mentioned in section 6.2 that Tenk is also using EP as strategy for the development of its products. Since Tenk is developing mobile applications and from literature review we also know that most of the mobile applications come under customizable generic products (as discussed in section 5.3). From our research we came to know that EP is very good for customizable generic products, therefore we can say that EP is very good to use for Tenk. The other reason for using EP at Tenk is its support for early acceptance testing of mobile applications. Another minor factor is the easy adaptability of EP for newly set up organizations. Applicability of EP has been discussed in detail in section 5.3. The results of the developed projects using EP at Tenk explained in section 7.4 suggest that EP is very good to use at Tenk.

8.2. Conclusions

The main aim of our case study was to make an initial validation of EP for customizable generic products by developing a real time customizable generic product (s) in industry. In order to achieve this aim we joined research and development department of TAT AB called Tenk. Furthermore, we also studied literature on prototyping and EP from 1991 till now.

The main rationale for the selection of TAT was its support for prototyping oriented development. Literature study was conducted in order to find current state of art and practices in the area of prototyping and EP. The literature study also enables us to have good grasp of EP oriented development. For this study we developed two real time customizable generic products using EP at Tenk. The two products are named Social Playlist and Social Location. Due to the limited amount of time in a master thesis work, we developed three prototypes for each of these products. Social Playlist was built was by six people including the authors of this study. And in Social Location product four people were involved including authors. For authors the main motivation behind joining the development team was to observe the process of EP. Another reason for authors to join the development team was the data collection from the applied process of EP. Being a member of the development team makes it easy to collect the data instead of other techniques like interviews, survey etc.

This case study identified situations where EP is a better choice as compared to the other approaches of software development. We also discussed advantages and disadvantages of EP and this will help in making trade off decision. Disadvantages of EP will help in analyzing the situation where it is not a good option. Results produced in this case study are not general because it was conducted in a single company. To make use out of the results company’s environment, nature of product, and its process are also mentioned. Our results suggested that EP is a suitable choice when company wants innovation, to clear ambiguous and sketchy requirements, to discover new requirements and to involve customer for his/her satisfaction and software testing. We can also see from the case study results that documentation and quality of a product is very weak in EP.

8.3. Our Findings

During our study we found some interesting findings from literature study and from our work at Tenk. Our findings from the conducted case study are as follows:
8.3.1. EP as a Concept

Most of the people think EP as a complete software process model. This misconception was also with us before starting the case study. In many literature resources we also found that EP was defined as a software process model. But we found few literature resources that were treating it as a concept that can be supplemented by some other software process model with iterative nature [24, 63]. This fact was also confirmed from our work and experience at Tenk. We found that Tenk was also treating it as a concept. Tenk was not applying it as a separate process model for its product development. Rather they were achieving this concept with the support from Scrum model that is iterative in nature.

8.3.2. Good and Bad of EP

Like all other software development approaches EP also has some positive and negative aspects. In our case study we highlighted both of these aspects by discussing situation where it could be a better choice, with its advantages and disadvantages. We also discussed why it is a good choice for the customizable generic product (see section 5.1 – 5.5).

8.3.3. An Initial Validation of EP for the Customizable Generic Products

In literature we found numerous case studies that were discussing the validation of EP for market driven and bespoke products. To the best of our knowledge, this work is the first case study that is discussing validation of EP for customizable generic products. Our results show that it is a good choice when you want to have innovative product, clear ambiguous and sketchy requirements, discover new requirements, save resources of software testing, involve and satisfy customer. It shows vulnerabilities in documentation of product and quality of code.

8.4. How this Report Can Benefit TAT AB and Tenk

Tenk was initiated by TAT AB as its research and development department. The main motivation behind the establishment of Tenk was to prototype an idea that can evolve as a future product for TAT AB. This will give an edge to TAT AB to compete with its competitors by introducing innovative product. Findings from this case study will provide more insight on EP for the practitioners at Tenk and TAT AB. Furthermore findings from this case study show that it is a good approach for Tenk as it shapes an idea to a prototype. In the beginning of Tenk, Tenk was interested to know that which software process model will be better for the development of their prototypes from an idea. But now findings from this case study will help them to change their focus from the selection of process model to the development of prototypes. This will enable Tenk to direct their energy only for prototyping of their ideas.

8.5. Future Work

We have identified some possible direction for future work within this area. These areas are as follows:

8.5.1. Validation of EP for Large Sized Customizable Generic Products
Due to scope of master thesis, it was only possible for us to make an initial validation of EP for small sized customizable generic products. Results and findings reported in this case study might differ in case of applying it for large customizable generic products. It is our opinion that if a case study is conducted on the selected approach for large customizable generic products, then it may find differences and similarities with our reported results in this case study. This kind of work will further validate the results of EP for large sized customizable generic products.

8.5.2. Validation in More than One Company

We performed initial validation of EP for customizable generic product in one company. Therefore, we cannot say the reported results are general and will suit to some other company. We can generalize these results only by conducting this kind of case study in multiple companies having different environment, processes and nature of product.

8.5.3. Comparing Other Approaches with EP for Customizable Generic Products

Based on this case study we cannot conclude that EP is the only good approach for developing customizable generic products. There may be some other approaches that can produce better results than the selected approach. This can only be explored by conducting a case study on different approaches applied on same product. By doing so, a comparison of EP with other approaches can be made and will most likely highlight more its strengths and weaknesses.
9. REFERENCES


http://www.agilemodeling.com/essays/fdd.htm


http://www.ub.es/multimedia/iem


