Surveying and evaluating tools for managing processes for software-intensive systems

Anuradha Suryadevara
IDT - Mälardalen University, ABB
Supervisors: Peter Wallin, Stig Larsson
Examiner: Frank Lüders
Abstract

Development of a software product consists of life cycle processes for different activities such as planning, developing and integrating software components. Organizations adapt different process methodologies to develop a software product. To manage these processes, tools are continuously introduced to develop well-defined processes to help organizations achieve their goals and business objectives on time, and with better quality products.

In this thesis, we evaluated two process authoring tools namely IRIS Process Author (IPA) and Eclipse Process Framework Composer (EPF Composer) for their suitability to support ABB software development processes. To capture the existing process, a series of interviews were conducted with development teams and process integrators. An extended evaluation criteria, based on Kellner’s requirements framework, was considered to evaluate process modeling aspects of the tools. Finally, the evaluations results were discussed and suggestions were made on the usability of tools to improve the organization’s integration processes in particular. IPA is validated as a suitable tool for process modeling of ABB software development activities.
Preface

I would like to thank my supervisor Peter Wallin for dedicating much time in supervising and providing assistance in conducting case study. Peter was always accessible for providing useful guidance in right direction. I would like to thank my (industrial) supervisor, Stig Larsson for giving me the opportunity to work on this thesis. His guidance in general and in particular his research insight into software product integration process has provided critical inputs to the thesis. Special thanks to my thesis examiner Frank Lüders for providing assistance when needed.

I would like to thank ABB and participants in interviews for their cooperation as well as valuable inputs. I would like to thank all my friends at Mälardalen University for discussions and support.

Finally, thanks to my family for all the inspiration and support.
# Table of Contents

1. Introduction  
   1.1. Software Engineering  1  
   1.2. Software Lifecycle Models  
      1.2.1. Code-and-Fix Model  2  
      1.2.2. Waterfall Model  2  
      1.2.3. Spiral Model  3  
      1.2.4. Agile Methodology  4  
   1.3. Software Development Processes  5  
   1.4. Software Processes and methodology Frameworks  6  
   1.5. Thesis Overview  7  

2. Problem Analysis  8  
   2.1. Problem Description  8  
   2.2. Method of Approach  8  
      2.2.1. Process Data Collection – Interview Procedure  8  
      2.2.2. Validity  10  
      2.2.3. Process Description  11  
      2.2.4. Tool Evaluation Methodology  11  

3. Requirements Collections for Software Process Modeling  12  
   3.1. Software Process Modeling and Benefits  12  
   3.2. Basic Process Elements  13  
   3.4. Deriving Requirements for Process Modeling  14  
      3.4.1. Communication  15  
      3.4.2. Reuse  16  
      3.4.3. Evolution  16  
      3.4.4. Management of Processes  16  

4. Process Documentation, Tool Evaluation, Results  17  
   4.1. Process Documentation Using Tools  17  
      4.1.1. Tool 1: IRIS Process Author  17  
      4.1.2. Tool 2: Eclipse Process Framework Composer  21  
   4.2. Tool Evaluation Framework  23  
   4.3. Evaluation Results  24  

5. Integration Processes  27  
   5.1. Issues and Challenges  27  
   5.2. An Efficient Product Integration  27  
   5.3. Problems in Organization’s Product Integration and IPA  28  

6. Conclusions and Discussion  30  
   6.1. Limitations and Future work  30
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>References</td>
<td>31</td>
</tr>
<tr>
<td>Appendix A. Interview Questionnaire</td>
<td>33</td>
</tr>
<tr>
<td>Appendix B. Snapshots of Process Authoring and Publishing using IPA</td>
<td>35</td>
</tr>
<tr>
<td>Appendix C. Snapshots of Process Authoring and Publishing using EPF Composer</td>
<td>43</td>
</tr>
</tbody>
</table>
Chapter 1
Introduction

1.1 Software Engineering

Developing large, complex software systems need to take a lot of efforts because of involvement of high cost, risk of failure, long development time and continuous changes in requirements. The success rate of software development projects is low compared to other engineering domains, for example civil engineering and aeronautical engineering. The reasons include lack of planning for step by step development through definite milestones and deliverables, weak requirement analysis, lack of good review process, poor skills of developers in adapting to new technologies and tools. Also issues like incorrect estimation of cost and time, insufficient collaboration with the customer etc. Sometimes, the same processes that were used for mechanical engineering domains are used for software development.

Software engineering involves systematic practices for developing software [23]. These practices describe tools and techniques to design, construct and maintain large software products in an efficient, reliable and cost-effective way. The discipline involves data management, algorithms, programming languages, human computer interfaces etc. In ever changing global environments, new technologies emerge constantly and there exist an increasing need for creating innovative and high quality software products to fulfill business demands. Large projects involve different kinds of people with specific skills for contributing towards the projects. Their work needs to be continuously reviewed to ensure the quality. An engineering approach to develop high quality software includes guidelines such as correct estimation of cost, efforts, work flow management, defining clear objectives and deliverables, periodic reviews for quality control.

1.2 Software Lifecycle Models

The term lifecycle describes the software product development phases and its maintenance. Though the notion of software process exists in every stage of development, it is difficult to define correctly. Recent developments and research efforts has created an identity for software processes. The software lifecycle defines the states of a product as it passes from the initial state (user requirements) to the final state (product ready system). Lifecycle model is an abstract representation of the software lifecycle. For a software product, there can be different life cycles. For example, if the user requirements are well understood and invariable, then a shorter lifecycle with the phases such as clear requirement specification, design and implementation are sufficient. On the other hand, if the user requirements are varying then the complete requirement specification is not possible at the beginning. In this case, alternative lifecycle models such as prototyping, incremental and agile may be more suitable for the development. Thus, a lifecycle model represents specific kind of process instances.
Though software lifecycle and software process are closely related (and can be inter-changed), distinct views on them are discussed in [5]. Many lifecycle models are developed to structure, and describe the process of software development. There exist many lifecycle models in software engineering. In practice, development teams adopt to a combination of these models for a given project. The models provide a fixed generic framework that can be tailored to a specific type of a project. Example lifecycle models are **waterfall, code-and-fix, spiral, rapid prototyping, unified process (UP), agile methods** and **extreme programming** (XP). An overview of some of these models is presented below.

### 1.2.1 Code-and-Fix Model

It starts with an informal general product idea and continuously develops code through code-fixes until the product is ready. This model is the basic model used in early development of software and is suitable for small projects with well-understood requirements, resources and no maintenance needs. In the case of large projects, the encountered problems with code fix model are discussed in [1]: coding and fixing the bugs repeatedly leads to poorly structured coding; though the software is well designed, it may not match the user needs; it will be expensive to redevelop the software to meet the user requirements; further, it will be costly to fix the code without planning for testing and maintenance. A schematic view of the code and fix model is shown in Figure 1.

![Figure 1. Code-and-Fix Model](image)

### 1.2.2 Waterfall Model

The first formal description of the waterfall model is presented by Winston W. Royce in [2]. It is a classic process model that defines a sequential development process and is suitable to manage large software systems efficiently on time within the estimated cost. The model is developed from experiences in the development of software packages for spacecraft mission planning, commanding and post flight analysis. A schematic view of the model is shown in Figure 2. The basic version of waterfall model describes the ordered set of activities or phases flows from top to bottom. In later versions, it allows the iterations between the successive phases and supports documentation, at each stage, which is useful for communicating between designers, manager and customers. The documentation serves as specification and guides the test process, design modifications and code review. Further, to eliminate development risks, Royce proposed many improvements over the original model in [2]. Though the model is still considered as suitable for large software development projects, the model suffers from the
limitations of sequential development i.e., it is expensive to fix the software if any problems are identified in the later stages.

Figure 2. Royce’s original waterfall model [2]

1.2.3. Spiral model

The spiral model is an iterative model with risk analysis and risk management phases and is suitable for large and expensive projects where the iterations are planned 6 months to 2 years long [1]. This model addresses the problems in waterfall (as mentioned in 1.2.1.) and other previously defined models. However, it can accommodate the previous models as the special cases. While waterfall model is sequential, the spiral is an iterative model. As shown in the schematic view in Figure 3, the process starts at the center and iterations progress in the clockwise direction. The model is based on the following concept:

“Each cycle involves a progression that addresses the same sequence of steps, for each portion of the product and for each of its levels of elaboration, from an overall concept of operation document down to the coding of each individual program”

This model is developed to fulfill the goals of risk analysis and risk minimization. To accomplish this, each cycle identifies some of the objectives such as

- Objectives of portion of the product are elaborated for each cycle (such as functionality, performance and changeability)
- Alternative plans for implementing the product
The spiral model has got many advantages over the previous models such as code-and-fix model, waterfall model. The developer can customize the model for a particular software product. However, there are some difficulties related to required expertise in risk management and the need for elaboration of the steps in model. Problems at high risk are given priority for elaboration in next cycles. Developer’s expertise is necessary to identify the high-risk problems. For example, when a developer is implementing the design developed by an experienced designer, he may neglect to consider some important details. Hence, well-trained and experienced people are necessary to make spiral model successful. In contrast to waterfall model, the spiral model does not provide much documentation, and the developers need to communicate to avoid misunderstandings in interpreting the project requirements.

Figure 3. The Spiral Model[1]

1.2.4. Agile Methodology

Agile methodology is a collection of software development methodologies and techniques for reducing the development cost and suited for shorter development lifecycles. Today, many organizations are interested in adopting agile software development methods as they offer tangible benefits over the
traditional methods. The agile manifesto [4] was published in 2001 to define the approach for agile software development. It emphasizes: individuals and interactions over processes and tools; working software over comprehensive documentation; customer collaboration over contract negotiation; and responding to change over following a plan. There exist 12 principles to be followed behind the agile manifesto. Based on these principles there are many development methods like extreme programming, scrum, agile modeling, open unified process etc.

In traditional models, the users give the requirements in completeness before the product is developed. But this does not usually happen in the software development process. Customers impose changes continuously and also sometimes the requirements might be vague and hard to understand that developer’s skills are not sufficient to interpret them correctly. However, accommodating changes in requirements is highly expensive in software development in particular in later phases towards implementation. So it is identified that the continuous customer/user collaboration is necessary in product development activities to provide the required features.

1.3 Software Development Processes

While a software lifecycle centers on the software itself, defining the states through which it passes from initial state to the final state, a software process centers on the corresponding development process, through the set of activities carried out in a particular order, to develop and maintain the software system [5]. An organization can define its processes to accommodate their specific requirements. These activities include both technical and management issues. There might be different views within a process: for example, one view may define the roles involved in each activity of the product development while another may define the relation between those activities. A software process model is an abstract representation of a software process. Different process models represent different process approaches.

The lowest granularity of a process consists of development tasks or steps. A process step is defined as follows [6]:

- Each step has a well defined objective
- Requires people with specific skills
- Takes specific inputs and well defined outputs
- Well defined preconditions and goals to enter and exit the step
- Uses specific techniques and tools, guidelines and conventions to reach the goals of the step

Each step in the process is assigned a specific development time, resources and efforts by the project management. And in return it produces some output after each step. The information produced after each step might be helpful for the management for corrective actions to revise the efforts and resources. For example, the number of people involved with a particular step may need to be increased. And each step is validated by the associated review process. For example, if the purpose of the step is to design the software, then it is verified and validated using inputs and outputs of the step. While verification concerns with whether the output is consistent with the input or not, the validation is
concerned with whether the output is consistent with the user needs. Multiple steps carried out in a single software development process. The step should satisfy the following characteristics:

- Each step in the process has a precise meaning about what to be done, when and how.
- It must be predictable with respect to effort, cost, quality and success, so that the step or the process can be repeated in other projects of similar complexity.
- Predictability of the quality in terms of expected defects and the overall performance.
- Process should support testing and maintainability i.e. to ensure that the software will serve the user and maintainable by following proper development standards and documentation provided.

In the software development projects, while development phase incurs estimated costs, the defects in the delivered software often require unpredictable costs during its maintenance. So it is critical for the software development organizations to adopt an effective and useful process to produce defect-free software. Also, a well-defined process facilitates proper monitoring, diagnostic data for the project management.

1.4 Software Processes and Methodology Frameworks

For large development organizations, there are different types of software processes related to both production and management perspectives [10]. The production processes related to the construction and maintenance of the product and the processes related to the resource management (people, time, technologies) are management processes.

Example processes in an organization are,

- Process for development of software
- Process for project planning and management
- Process for configuration and management
- Process for product integration
- Process for managing software processes itself because the processes need to adopt to ever-changing technologies and development needs.

There exist different kinds of process methodology frameworks to structure, plan and control the process of development. These include Rational Unified process (RUP), OpenUp, Unified Process, Microsoft Solutions Framework, Agile methodologies etc. Based on the factors such as technical, organizational, nature of the project and team size, one of these methodology frameworks can be used by the development team for the development process. Software organizations follow process standards such as CMM, ISO12207 etc for developing software. These standards have well defined procedures that help organizations to improve their abilities to complete large projects on time with in the estimated budget.
To support authoring and customization of software development processes, OMG (Object Management Group) provides software process engineering meta-model (SPEM) specification. It is an open industry standard for modeling software development processes [14]. The SPEM 2.0 (latest release) meta-model consists of a minimal set of process modeling elements to describe software development process independent of specific process models or disciplines such as project management or analysis. But, because of its high complexity, it would be difficult for process engineers to use its concepts directly. To ease its complexity, a tool support is required [28]. And the process engineers would benefit from using the tools based on standardized process modeling language: easy to model the company’s existing software development process; software team members would have consistent and shared language across the organization; and additionally help to learn about current software industry best practices, customize and reuse. Process data can be interoperable among different SPEM compliant tools. Many open source and proprietary tools are developed based on SPEM, for example Eclipse Process Framework Composer (EPC) [16], Osellus’s IRIS process author (IPA) [15], IBM’s Rational Method Composer (RMC) [17], Softeam’s Objecteering [27] as well as many more available in project planning and estimation, resource management etc.

1.5 Thesis Overview

This thesis is organized into six chapters including the introduction chapter. Chapter 2 starts with the problem description, the method adopted for data collection, process description and the tool evaluation methodology is described. Chapter 3 includes the requirements collection procedure. It begins with the software process modeling objectives and benefits, basic process elements, short description of an OpenUp approach to software process modeling and at the end it contains the section on deriving the requirements for process modeling using the case study data. Chapter 4 presents the overview of the two considered tools for evaluation, evaluation procedure and the obtained results. General observations on software product integration processes, problems identified in the organization’s software product integration processes and the suggested solutions and the evaluated tool support for its improvement are presented in chapter 5. In Chapter 6, conclusions and limitations of the thesis work are presented.
Chapter 2
Problem Analysis

This chapter includes problem description of this thesis, method of approach, case study procedure and its validity. It is also discussed on how the process descriptions are defined and a tool evaluation methodology is proposed.

2.1 Problem Description

With growing need for quality software, organizations adopt standard process methodologies for its development. To organize the set of related development activities, it is essential to define a controlled development process. There exist standard methodologies such as OpenUp, RUP, Unified process, Agile methodologies etc. However, these methodologies need to be customized for the software domain, both the development phases and the organization itself. Still, adapting to a specific lifecycle is not sufficient to guide and control a software project, a software process provides concepts to frame and organize different issues related to development activities [26]. The main focus of this thesis is to evaluate different process tools based on their technical suitability for software development process automation applications at ABB. This requires identification of applicable process methodologies and related tools. Also, other objective of the thesis is to make general observations about problems in software integration process as well as to discuss how ABB can make use of the evaluated tools for its software integration improvement. In this thesis, EPC and IPA are considered for evaluation based on the evaluation framework given in section 4.2.

2.2 Method of Approach

In addition to conventional solutions, specific field study and experience data is required. These aspects of the problem are investigated through three phases: process data collection, process description, and tool evaluation methodology.

To capture the involved process activities and other aspects of the development methodologies, a suitable investigation approach is required. For that, interview case study is adopted as appropriate method. From the process data, different process elements were identified and the process descriptions were documented using the considered tools based on process modeling objectives [13]. Further a tool evaluation framework is needed for comparison of evaluation of considered tool environments for process descriptions. In this thesis the tool evaluation framework is derived from the Kellner’s requirements formulism for process modeling, tool interface and vendor aspects and some general aspects such as cost, time to adapt to tool environment.

2.2.1. Process Data Collection – Interview procedure

To obtain various aspects of the software development process, an interview method was adopted [7]. In consultation with domain and process experts, a suitable questionnaire was prepared. A set of three
interviews are planned and conducted. The data is collected in predesigned tables. No recording devices were used in order to avoid the discomfort for the respondents while providing their insights and also to avoid distractions during the interview itself. The data collected from the interviews was analyzed through the activities captured and compared with the activities defined in OpenUp software development process methodology. Different artifacts and documents were identified from the source of collected information.

The questionnaire focused on the development activities in general and specific to the integration phase. The questions were divided into seven sections based on software development phases and also include organizational and general aspects. The questions are structured into related sub questions. The more generic questions are in the first level. The detailed questions are in the next level; those could be helpful for the respondent to understand the question in right manner i.e. to avoid the risk of misinterpreting and to provide his/her own insights into situation without deviating from the line of inquiry. All the interviews were conducted using the same questionnaire in the consistent order of questions. The questionnaire was divided into seven sections related to organizational, project planning, project architecture and design, handover from development to integration, integration, testing and also some general questions. The questionnaire is attached in appendix A.

**Organizational**: to investigate the details of respondent’s experience with the company, previous experience, duration of projects, whether the projects were in-house or distributed and the respondent’s role in the concerned projects. These set of questions were helpful to make the respondents comfortable and to motivate them for their interactive participation in the interview.

**Project Planning**: aimed at understanding project design process. It is intended to capture the practices followed and who is responsible for the overall design of the project.

**Project Architecture and Design**: to collect the details how the architectural design process was usually carried out, associated artifacts, the people (or their roles) involved in the process, key measures taken during the process, and handling the critical situations like change in requirements.

**Handover from Development to Integration**: as the purpose of the interview is to capture the product development activities, roles and the artifacts involved at each phase, the specific questions in this section deal with:

- the required quality check for the developers work
- responsible supervisors at this phase
- artifacts involved as input, and output to activities
- steps involved and validation of the developers work

**Integration**: as in conventional software development product lifecycles, integration phase is close to that of implementation. The related questions are aimed at investigating the specific steps involved in integration, handling the common but critical problems like delays in sub-modules, measures taken when the integration is failed, and identifying the propagated technical problems from earlier activities. Other aspects investigated are related artifacts, persons (or roles) responsible for integration etc.
As integration phase is one of the main aspects of the focus of this thesis, detailed questions were aimed at finding causes of integration failure, associated time factors and the responsible person roles. As described above, the questions are in two levels, the specific question in first level and the general questions are in the next level (a, b, c etc) as shown below.

Integration: can you describe the integration process?

  a.) What are the different stages in software integration process?
  b.) How do integrators manage when the modules are not delivered on time?
  c.) How do you trace the problems like build is not successful on integration due to incomplete updating of the repository?
  d.) What are the responsibilities of a developer at this stage?
  e.) How the hardware is integrated?
  f.) Integrated build is executed manually or automatically by a tool?

Testing: to investigate the test activities throughout the development phases. This includes identifying specific process activities and related steps for conducting the system test, responsible persons and the related artifacts (documents and reports).

General: for identifying and documenting general explanations by the respondents related to activities, responsibilities and the involved risks. Further, the respondents were invited to suggest changes in their development activities for the process improvement. In a conclusion, the final question was included to elicit the respondent’s opinion on introducing new process tools.

A trial/pilot interview was conducted to work with the correctness of the questionnaire. This was also helpful to gain an experience in different aspects such as data collection, establishing informal and casual environment among the participants for promoting the success of the case study. Two/Three respondents were interviewed for a short period of time about one to two hours. The respondents chosen were responsible for coordinating the team, delivery of subsystems or the delivery final system. There were a number of nine people altogether, include project manager, product manager, configuration manager, line managers and integrators.

2.2.2. Validity

In this thesis, four types of validity are considered: construct validity, internal validity, external validity and reliability [7].

The construct validity: construct validity is related to the data collection procedures in this case study. The sources of evidence used in this case study are interviews and the organizational document with the relevant data. The interviews conducted under guidance of the thesis supervisor to ensure the quality in the interview process and the relevant data collection. The questionnaire was reviewed by the domain experts and a pilot interview was conducted.

The internal validity: the data collected from the case study is related to the software development activities and the problems in integration processes. At the time of data collection, all the case study
respondents were involved and responsible for software development activities. Hence the collected data is considered as relevant and valid.

**The external validity:** since we aim to capture a process that is internal to the company, it is hard to generalize. But the questionnaire is based on the software development activities in general, not in particular to the domain specific software development. Hence, the issues found in the interviews might be applicable to any software development domains as well.

The reliability: the case study can be repeated by another investigator either to reach the same conclusions or to improve the procedure. The reliability aspect of the case study is addressed through the detailed description of the procedure used as described above and the questionnaire attached in Appendix A.

2.2.3. Process Description

With the data collected from the interviews as described above, basic process elements such as roles, activities and work products were identified. Two different SPEM-compliant process author tools were considered for process description. However, as the organization under investigation was using a combination of waterfall and agile process methodologies for its product development, **OpenUp process methodology** was chosen as the most suitable one. And the identified process elements and relations are documented in the tools.

2.2.4. Tool Evaluation methodology

In this thesis we considered a bottom-up approach to derive requirements framework for evaluation criteria to validate tools for their suitability in the chosen domain. Here bottom-up approach means considering user perspective of the process in deriving the requirements, as proposed in [13]. The requirement formulism, defined in [13], is intended for process modeling. But it can also be imposed on process modeling tools. In the tool evaluation criteria, apart from the list of requirements described in [13], two more requirements were considered: user interface, and vendor support. It is shown in Table 1. Further some requirements may be derived from the experiences gained in documenting the process in the tools.
Chapter 3
Requirements Collection for Software Process Modeling

This chapter introduces the concepts of software process modeling and benefits, basic process elements required for process modeling. An example software process, OpenUp is discussed briefly. At the end, requirements for process modeling are derived.

3.1 Software Process Modeling and Benefits

This section describes software process modeling, its benefits and a brief summary of basic process elements.

Software process modeling deals with creation of software process models [10]. The process models are used to analyze, understand and improve the specific processes. The process engineers in an organization are responsible to develop flexible and customizable process models that are interoperable with the tools of the development environment. Kellner [13] identified four objectives for a successful implementation of software process modeling. Further, a set of requirements for a representation formulism are derived from the objectives. In relation to [13], Curtis et.al defined some of the objectives and benefits of software process models [11] are given below.

**Ease of understanding and communication**: sufficient information is required to represent a process model. This formal definition of the process provides a basis for training.

**Process management support and control**: through project specific processes, process monitoring, management and co-ordination.

**Provision for automated orientations for process performance**: by providing user orientations, instructions and user manuals for effective software development environment.

**Provision for automated execution support**: through process automation and process integrity assurance.

**Process improvement support**: requiring process reuse, enhancements through comparison with related processes.

Software organizations improve development efficiency through well-defined development processes. However, developing software process definitions for large organizations involve high cost. Thus the process definitions should be designed and developed for general purpose within the organization. It facilitates reuse of the process definitions for common activities of different projects within the same organization. These generic process definitions can be tailored to different project specific purposes in the organization by making necessary changes [9].
3.2 Basic Process Elements

There exist two inter-related software processes in any software development organization [10]. Product processes: related to the product development and maintenance activities. Management processes: related to resource management for effective control of the production processes. The commonly used elements for process modeling are activities, products, resources and roles.

Agent (or) Actor: An actor is an entity that executes a process. It can be a person or a system. Human actor refers to the people who involved in developing software or a process. System actor refers to the computer software or hardware. Actors are characterized by roles. An actor can play one or more roles, performs a set of activities. Also, more than one actor can be involved in a role.

Role: It represents a set of responsibilities, i.e. it describes the required skills to perform a specific software process activity.

Activity: produces externally visible changes in the state of a software product. An activity is performed by a set of role(s) and also produces related artifacts (outputs).

Artifact: An artifact or the (sub) product is the raw material of a process. Artifacts are created, accessed and modified during the process. An artifact produced by a process, later can be used in the same process or in another process.

Figure 4 shows the relationship between actor/role/activity, activity-activity, product-activity and product-product. The relationship composed of indicates one or more occurrences of corresponding element. For example a set of products (software artifacts) are delivered to the customer as single product. i.e. a composed of relation exists among the set of products. The interrelationships between these basic process elements are discussed in detail in [12].

![Figure 4. Basic Process Elements [10]](image)

3.3 Software Process Modeling – The OpenUp Approach

As many in-house projects are developed through a combination of incremental and iterative processes, OpenUp [16] process methodology is considered as a suitable approach for these kinds of projects.
OpenUp is an open source process framework which combines the RUP [17] as well as light weight agile practices [1]. It is developed as part of Eclipse Process Framework (EPF) [16]. This framework supports an incremental and iterative software development approach. The OpenUp lifecycle is structured into four phases: Inception, Elaboration, Construction and Transition. It promotes cooperative and productive team environment i.e., team collaboration and coordination to understand the project; gives priority to stakeholder demands by involving stakeholders in the project development.

In OpenUp, the project architecture is central to risk minimization and overall organization of the project. Further, OpenUp facilitates continuous feedback from the customer to improve the project development. OpenUp defines a set of small process elements that can be combined to describe various end-to-end processes for different but related projects within the same organization. The set of process elements and their relations are shown in Figure 5.

![Figure 5. A schematic view of OpenUp elements – roles, work products, tasks, and their relationships](image)

### 3.4 Requirements Derivation for Process Modeling

Though the OpenUp framework is found suitable for modeling the in-house process, the data collected is insufficient to fit into the all phases of OpenUp process. The goal of this thesis is not to define the complete process, but evaluate the considered tools for their technical support for the organization’s software development process. Thus the process is completely modeled from the start using the data collected from the interviews. To develop the process Kellner’s representation formulism is used as a basis [13]. These requirements are imposed on the process modeling objectives and on the tools used to develop the process. Further, the case study information was considered as important in designing of the process. Especially, for documenting the organization’s existing process in the tool, the data was collected from the experiences of software developers and managers involved in the software
development activities in the organization. However, none of the case study participants had any experience in working with a process tool or any specific process methodology. Therefore it was important to gather different opinions on adapting to a process methodology and related tools.

Basically, the derived requirements for process modeling are categorized into four divisions that are based on its objectives.

3.4.1 Communication

Communication aspects focus on effectively communicating the description of a process to others, such as workers, managers, and customers. Process models are especially useful for sharing knowledge and expertise; for example, process descriptions can be used for training purposes. These aspects can be formally grouped into specific requirements as below.

**Visual approach to information representation:** visual representation of the information such as figures and tables should be used. A graphical representation can convey considerable amount of information.

**Multiple and complementary views of a process:** to facilitate different types of process users, process descriptions should be viewed in different perspectives. Most of the users need to know the process details of their concerned activities in the project. Kellner’s requirements address different viewpoints of the process and various approaches on their abstraction are identified.

**Enable process communities and collaborative process framework:** process repositories allow process communities. Process communities facilitate their members to consume and provide the feedback on an organization’s process. Process repositories do not only contain the process models but also provide information about previous experiences and best practices. Process users are the important source for identifying the problems in the existing process. And the lessons learned from their previous experiences can be useful for the process authors to improve the process. Process managers can make use of this information during the execution of the process to select upon the different versions of the processes.

Providing a collaborative framework means, both the process users and process authors can contribute to improve the process. However the changes are not automatically edited, finally the process engineers edit and update the processes with the provided feedback by the community members.

**Define syntax and semantics for process descriptions:** process descriptions should be constructed based on clearly defined syntax and semantics. This facilitates analyzing and executing the process descriptions. Predefined syntax and semantics for process modeling define a meta-model. SPEM is a meta-model for defining processes and its components. It defines a set of modeling elements to describe any software development process. It also describes the structure of the process repository which stores the process models. Defining a meta-model offers an automated tool support for process authoring and customization. SPEM supports many process tools. For example Eclipse process framework composer and IRIS process author. It imposes some validation rules to check for the validity of the formal process descriptions and processes. Process models created in one SPEM compliant tool
can be imported from or exported to other SPEM compliant tool to eliminate the risk of relying in just one vendor.

**Different formats of the process:** when a new project is planned, a process is set for its developmental activities. A process involves various types of stakeholders such as customers, project managers, developers, quality managers and so on. Different stakeholders have different demands to access the process. Customer needs a printed copy of the process to check for the deliverables and milestones. And the developers need to browse the process to capture the information for their project developmental activities. The managers are interested in the information related to resources planning such as cost, time and manpower.

### 3.4.2. Re-use

**Process tailoring:** every software process is unique. And the developmental process would differ from one project to the other. Creating a new process usually involves high cost and takes a lot of time. However, at times, processes might be tailored to fit new projects. Keller’s second objective for process modeling is to facilitate the reuse of the process. That imposes the process to be customizable or tailored.

### 3.4.3. Evolution

**Process modifications and alternatives:** process evolution occurs in different situations. Process descriptions should be formally validated to check for end-to-end work flow. And the process descriptions support the information such as project duration, manpower and quality measures of the project in development. These attributes in turn cause modifications in the process descriptions before executing them. Once the process is in use, the process communities facilitate its members to provide the feedback. This would also cause for process modifications.

### 3.4.4. Management of Process

**Synchronization and version handling:** version handling requirement should provide support for creation and management of various versions of Process descriptions. This should facilitate reuse of the portions of project descriptions available, and compare the process descriptions of different versions. Synchronization facilitates multiple users to work on a process simultaneously.
Chapter 4

Process Documentation, Tool Evaluation, Results

This chapter is mainly focused on the process documentation, tool descriptions and the evaluation procedures. Finally, the tool evaluation results are discussed.

4.1 Process documentation using Tools

Two types of inputs were considered when deriving the elements to model the (ABB) software development process.

- The data collected from the interviews, which include user experiences in software development activities, related artifacts, guidance etc.
- In-house documentation involving the roles involved in the software development projects.

From these two sources of information, the basic process elements such as roles, activities, work-products and their relationships were identified. Based on this, a suitable process is defined and documented with tools such as IPA and EPC.

4.1.1. Tool 1 – IRIS Process Author

Osellus is one of process improvement and automation solution providers for software industry [15]. It is a Canadian software company profiled by IDC [24] for its efforts in innovation, speed, power and flexibility in developing solutions. It is a member of Microsoft inner circle partner [25] and IBM’s ISV Advantage Program [17]. One of its products, IRIS Process Author (IPA), is a software process automation tool to help organizations to model, improve and automate their software development processes. While existing tools in the domain are usually capable only to automate certain tasks or specific phases of software development lifecycle (for example, project architecture and design phase), IPA supports end-to-end processes over entire software development lifecycle. Further, the tool is capable of addressing organizational concerns such as business requirements like time to market, product quality and cost. The features of the tool are described in detail below.

IPA is based on software process engineering meta-model, i.e., the SPEM specification of OMG [14], which addresses the software industry’s complex process requirements. It provides a common framework for modeling software development processes and their components. A minimal set of process modeling elements such as roles, tasks, artifacts, checklists and work products are defined in the specification to describe many aspects of the software development process. Further details about SPEM are available in [14] and [15].

As SPEM offers an open industry specification, the SPEM compliant tools support the process authoring and automation independent of specific product or technology. IPA is web based with a visual process management system. It uses web 2.0 technologies to provide a central portal for process communities and related activities. It provides a visual modeling interface that supports both simple and highly complex processes. Also, Osellus process libraries (OPL) contain a list of pre-defined processes for reuse
and adaptations. Further, using IPA, multiple projects can be mapped to the modeled process. This supports distributed teams working on different processes and methodologies.

**Process Authoring in IPA:** the process author creates process package to store the process contents. Process packages are stored in libraries. The user can create and modify the contents in the permitted libraries. In order to facilitate process reuse, the process content is structured into three types of process packages that can be created in the library: **packages**, **process components** and the final end-to-end **process**.

**Packages:** The process elements such as roles, activities, guidance, work products etc. These process elements are commonly used across multiple processes. The package element view is shown in Figure 6.

**Process components:** The common work break-down elements across multiple processes that are used to construct a workflow are defined in these packages.

**Process:** This package contains an end-to-end process workflow constructed from the process elements and workflows defined in the native package or imported from the common definitions of packages or process components.

![Process Authoring in IPA](image)

**Figure 6. Package elements view in Process Author**

When a large amount of process data is created with complex workflows, it would be a difficult task for the process authors to validate process manually. Process validation is a very useful feature of IPA to verify the created process data automatically. Validation procedure is based on the SPEM-validation rules (for example, there must exist an initial activity that starts the process).
Validation strategy differs depending on the type of process packages (as defined above). More details on process validation rules are provided in the tool documentation [15]. If the validation fails, the validation log displays the failure details where the process can be modified accordingly. The process package validation window is shown in Figure 7. Once the process is validated successfully, it can be published in different formats such as an online process or as a wiki site or at central repository called IRIS process central.

Figure 7. Validate Process Package

An online published process: an HTML based website can be generated. It can be hosted either online or offline on the local system which allows an easy access through exploring the process.

Published process as a wiki site: a wiki site can be generated on the hosted server, which facilitates the process users to provide their feedback on the published process.

Published process at IRIS Process Central (IPC): to publish the process in different formats (including online process and wiki) to the IPC. A snapshot of an online published process using IPA is given below in Figure 8. It is an online process repository and acts as an online portal to the organization to allow the authorized users to browse, re-use and provide the feedback on the processes. Also, it facilitates collaboration among process authoring and the process communities through blogs and discussion boards. The processes in IPC can also be exported in different formats including online website, portable document, wiki-pages and more. The information provided in the feedback is useful for the process
engineers to improve the existing process and also the managers would benefit in selecting suitable process (version) for enactment. Snapshots of process authoring and published process for ABB software development are given in Appendix B.

![Figure 8. A snapshot of the outline of published process with IPA](image)

The main features of the IPA tool and related observations are described below.

**User interface:** Drag, drop and connect the process elements to construct workflows, work break down structures and state diagrams etc. IPA facilitates creating and editing all the process information including descriptions, guidance and other fields. Rich text diagrams such as workflow diagram, work breakdown structure and work product association etc are used to visualize the process model in multiple dimensions.

**Process architecture:** Assigning access rights to the specific process libraries for valid users, creation of process packages, importing process packages within the process or across the processes, built in support to tailor the content of well known industry methodologies such as RUP, OpenUp, and Agile etc.

**Process definition:** Process elements are based on the definitions provided in SPEM. Process element fields captures the full details and is able to add customizable attributes to the process elements, links to the external guidance resources, work product state machine diagrams, assigning activity preconditions and goals on work product states, and application of SPEM based validation rules for automatic validation of the process for its integration assurance.
**Importing and exporting**: Importing and exporting process models to other SPEM compliant tools. XML based importing from other industry methodologies. And the define processes can be exported to MS-project (Microsoft project) and VSTS (visual studio team system) [25]. Compliancy reports can be generated by comparing with the governance frameworks such as COBIT, CMMI etc.

**Publishing and feedback**: The processes can be published in three different formats (details above). The output can be shown in word or HTML formats. Wiki pages facilitate the users and subject matter experts to provide their feedback on the process, which is useful for process tailoring and improvement. Publishing to process central enables online process communities and captures observations on enactment procedure and there are many search options to track the process versions.

### 4.1.2. Tool2 – Eclipse Process framework Composer

Eclipse process framework (EPF) is an open source project in eclipse. The project goals are set to provide a customizable software process engineering framework and example process content and tools. Eclipse process framework composer (EPC) is an exemplary tool platform for software process management provided by EPF. It runs on windows and Linux platforms. The tool is for process architects, process engineers and managers to author, tailor and publish methods and processes for different software development organizations supporting various kinds of projects and development styles. EPF provides three exemplary process frameworks such as OpenUp, Extreme programming and scrum, where OpenUp was the first delivered process. Nevertheless, it supports its user to create own process frameworks from the start. The descriptions and processes are based on SPEM specification. EPC obeys 80% of the SPEM specification according to EPF 1.5.1 Release Review, September 22, 2010., and aims to fully support SPEM.

EPF method composer library contains method plug-ins and method configurations. A snapshot of the tool editor containing method plug-in and method configuration are shown in Figure 10. Method content and processes are organized as method plug-ins. The tree structure of the EPF composer method library is shown in Figure 9. EPC clearly distinguishes method content and processes. It provides standard set of method and process definitions. Method content is used to define the basic process elements such as roles, tasks, work products and guidance and these elements are used to define the processes. The EPF composer supports two kinds of processes i.e. **capability patterns** (building blocks of a process) and **delivery processes**. The process author defines the method content and capability patterns before defining the final end-to-end process (called delivery process). Capability patterns are assembled into a delivery process which represents a complete project lifecycle. In other way, EPC provides standard set of reusable method and process definitions. It includes OpenUp and many plug-ins extending OpenUp to meet the specific project needs. The existing method content and process definitions can be selected and assembled into a method configuration to meet the needs of a particular type of development project in the organization. And also the user can extend the existing method content and processes to define a configuration that address the domain specific needs. The created method configurations can be published in the HTML website. Various publishing options will be used to customize the look and behavior of the published site. Snapshots of process authoring and published process for ABB software product development are given in Appendix C.
Features of EPF Composer and related observations are listed below.

**User Interface:** Rapid process assembly via drag and drop, different process views such as work breakdown structure, team allocation view and work product usage. It is possible to select, combine, tailor and assemble the process configuration from the method content for a project. Well structured user interface and intuitive rich text editors for creating and managing content descriptions. Textual and graphical representation is made of the method content.

**Process Architecture:** EPC is a document based system that can be well integrated to any version control system like CVS, and ClearCase. Distinguish method content and processes, manage content in libraries, reusable plug-ins and packages. EPC supports the processes that are based on different lifecycle models such as waterfall, iterative, and incremental lifecycles. Further, it is possible to select and tailor the content of the exemplary process frameworks such as OpenUp, Extreme programming, Scrum.

**Process Definition:** EPF Composer is partially SPEM compliant. Method content elements can capture the information with the providing element attributes, additional attributes cannot be defined. It provides well formatted rich text documentation that allows format text, work with tables and include images etc. It allows validating the processes by enabling method library validation (on/off).

**Importing and Exporting:** already created method content can be imported either by importing method libraries, method plug-ins or XML (depends on third party vendors to export the library content in XML format and uses import wizard to import XML files). Processes can be exported as XML files and open them in MS projects. And the method libraries can be exported using method configurations with all included plug-ins.
Publishing and Feedback: method configurations can be exported as a HTML website. And using a single feedback link available on a published site where users can provide their feedback. But it does not support more than one link for collecting feed backs on different entities like management, development etc. It depends on external sources like installing EPF wiki to extend this facility.

4.2 Tool Evaluation Framework

The requirements framework, as defined in chapter 3, was applied to the tool evaluation. In addition to those requirements, experiences gained from the tool usage, user interface, cost and vendor support are also included in the evaluation criteria.

The process modeling objectives, as described in chapter 3, are extended with tool specific objectives for its evaluation. A relevant tool objectives and a set of requirements imposed on them for efficient process modeling, tailoring and publishing are shown in Table 1.

Table 1. Process Tool Objectives and requirements

<table>
<thead>
<tr>
<th>Objectives</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>1. SPEM compliancy</td>
</tr>
<tr>
<td>(extended)</td>
<td>2. Flexibility</td>
</tr>
<tr>
<td>Communication</td>
<td>3. Visual approach to information presentation</td>
</tr>
</tbody>
</table>
4. multiple and complementary view of process
5. enable process communities and collaborative process framework
6. define syntax and semantics for process descriptions
7. Different formats of the process (HTML, PDF, etc)

| Reuse | 8. process is tailorable
|       | 9. integrate with other configuration management tools
| Evolution | 10. Process modifications and alternatives
| Management of processes | 11. Synchronization and version handling
| Vendor support | 12. Open source/proprietary
(extended) | General (extended) | 13. Cost, time, Operating System support

4.3 Evaluation Results

The evaluation results are described in details below. A summary of specific results are given in Table 2.

User Interface: this objective can be discussed in different aspects: technical and general. The meta-models of the tools are structured on SPEM based rules. The element attributes can be customized in IPA. For example, new guidance ‘type’ can be created by the user, new attribute to the role (or any process element) can be added etc. EPF Composer does not support the customization of attributes. Look and feel of the content is well organized in both the tools and any external sources of information like documents and templates that can be stored in their own formats. However, IPA provides a more user friendly interface compared to EPF composer (for example, simplified use of complex meta-model concepts). In IPA, drag, drop and connect over the elements to model workflows, state machine diagrams etc is more flexible.

Communication: in IPA, Rich text diagrams Such as workflow diagram, work breakdown structure and work product association help to visualize the process in multi dimensions. Also, work product dependency diagram helps to understand the process relationships and simplify navigation through the process. In EPF Composer, work breakdown structure, activity diagram (nested work flow representations) and work product dependency diagram serve the same purpose. In both the tools, the three diagrams are synchronized with the associated work breakdown structure. Any changes made in the diagram editor can be reflected in the process structure.

IPA provides an online process repository called IRIS Process Central (IPC) that enables process communities to interact and provide feedback on the organization’s processes. The processes can be stored in several formats such as, online websites, printable formats, wikis, enactment templates, compliancy reports (how the processes comply with governance requirements such as CMMI, COBIT
etc). A wide variety of search options are available for all type of process documents. Also, IPC can be customized using web share 3.0 technologies. Looking into EPF Composer, it does not support any repository for its processes. It can provide a link to collect the user feedback, which does not promote collaborative work between process users and process engineers.

**Reuse**: the processes created in IPA can be imported from or exported to other SPEM compliant tools. Processes imported externally are XML based. IPA allows tailoring the content of industry methodologies such as OpenUp, RUP, and Agile etc. IPA architecture supports the reuse of the organizational existing processes, through its inheritance model. The available processes can be tailored and customized to fit into project needs. EPF Composer facilitates process tailoring with the reusable process building blocks such as capability patterns and method content. EPF Composer supports using and tailoring the existing process framework such as OpenUp, Scrum, and Extreme Programming. EPF Composer cannot import the external method libraries by itself, and instead depends on third party vendor (for XML imports).

**Evolution**: IPA supports feedback mechanisms to allow the community members to browse, discuss and contribute to the processes through its process central. Lessons learned, best practices and compliancy reports are stored in IPC. Process engineers would make use of this information to modify the processes. Different versions of processes maintained in IPC help managers to select upon the project relative process during enactment time. EPF Composer does not provide any internal mechanism that supports process evolution.

**Management of processes**: to provide synchronization among different users of a process, IPA provides role based controlled access to the process libraries (but mentioned as packages on Osellus website, which really did not work during evaluation). IPA uses lock-modify-unlock mechanism to provide an access to the specific valid user. The user can lock the library as long as he/she is modifying the process. However, only one user can work at a time. EPF Composer depends on external revision control systems like CVS and Clear Case to allow multiple process authors working in parallel. When the method library is added to CVS/Clear Case, versions of various files in the library can be controlled allowing multiple authors to work in them simultaneously. For version handling and synchronization among multiple users, EPF depends on an external revision control systems (which store process content as text files). The capability patterns or the delivery processes can be exported to MS Project in XML format. The processes can be published as websites. In IPA, all the process descriptions are stored in external revision control systems in XML format for its version tracing. And the XML files can be imported back to IPA for publishing etc. The changes made in the process versions are automatically propagated into the repository for that derived processes i.e. process integrity is maintained. To provide support for different stakeholders, various formats of the processes are available such as printable documents, wikis, online websites etc. And the processes can be exported to project management tools like MS Project, VSTS.

**Vendor support**: IPA is a commercial tool, the maintenance and support is provided with the purchase of the tool (vendor details are discussed in section 4.1). EPF Composer is an open source tool, for which the user has to refer to its documentation and the organization has to take care of its maintenance.
**General:** cost and time to adapt to new tool environment should be considered in the tool evaluation.

**Table 2. Overview of evaluation results**

<table>
<thead>
<tr>
<th>Evaluation Aspects</th>
<th>IRIS Process Author (IPA)</th>
<th>Eclipse Process Framework Composer (EPC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface</td>
<td>o Simplified use of meta-model concepts&lt;br&gt;o Customization of element attributes</td>
<td>o Complex meta-model concepts&lt;br&gt;o Does not support customization of user interface</td>
</tr>
<tr>
<td>Communication</td>
<td>o Supports rich text diagrams for process visualization&lt;br&gt;o Enable process communities through its process central</td>
<td>o Supports rich text diagrams for process visualization&lt;br&gt;o Does not have built-in support to enable process communities</td>
</tr>
<tr>
<td>Reuse</td>
<td>o Supports to tailor content of many industry methodologies (details in 4.1.1.)&lt;br&gt;o Built in support for XML imports</td>
<td>o Supports to tailor the content of three industry methodologies (details in 4.1.2.)&lt;br&gt;o Depends on third party vendor for XML imports</td>
</tr>
<tr>
<td>Evolution</td>
<td>Through its IPC feature, documents feedback from community members, lessons learned, best practices, compliancy reports that help to process evolution</td>
<td>Does not provide internal mechanism to support process evolution (feedback can be collected through the provided link in the published process)</td>
</tr>
<tr>
<td>Support for Process modeling</td>
<td>o Provides support for synchronization among different process authors through lock-modify-unlock mechanism (controlled access)&lt;br&gt;o Version handling is supported by external revision control systems&lt;br&gt;o Export the process content in XML format to MS Project, VSTS and more&lt;br&gt;o Formats of published processes are customizable: wikis, online websites, printable documents and more</td>
<td>o Depends on external revision control systems for both synchronization and version handling&lt;br&gt;o Export the method content and process content in XML format to MS-Project&lt;br&gt;o Supports process publishing in HTML and PDF formats</td>
</tr>
<tr>
<td>Vendor support</td>
<td>Proprietary tool</td>
<td>Open-source</td>
</tr>
<tr>
<td>General usability</td>
<td>Intuitive interface and automatic validation supports user to adopt to the tool environment</td>
<td>Tool adoption requires much user efforts</td>
</tr>
</tbody>
</table>
Chapter 5
Integration Processes

This chapter discusses various issues and challenges in software product integration processes. Further, some conclusions are made on how well the tool IPA supports to eliminate the identified problems in the product integration phase of software development.

5.1 Issues and Challenges

Software Product integration process describes the process of assembling software components into more complex components or into a complete product. It ensures the product as integrated functions properly. There exist certain well-known problems in product integration [20]. The detailed views of these problems are discussed in [19]. An overview of these is presented below.

Problems related to architecture and design: product interface requirements, specifications and designs are done without considering the full system; the problems related to interface change issues are also reflected in product integration.

Problems related to inadequate use of integration environment: usage of different test environments at component level and integration level; inadequate preparation of build environment and changes made in the integration environment without proper verification.

Problems related to inadequate delivery of functions: code with incomplete functionality is usually delivered to integration. Incomplete updates of repository; files not included in the build as required/planned; the code functionality is poorly tested which leads to problems in making build or in the integration and system test.

Martin Fowler addressed the problems in software product integration specific to the projects using agile methods with distributed development [22]. Those relate to difficulty in specifying the semantics of an interface, integration test process, developers responsibilities in build environment, practices in using the revision control systems for code check-in and check-out, inappropriate configuration of source code control systems and also the problems related to the distributed teams among different countries.

5.2 An efficient product integration

It is necessary to define the scope and the integration strategy should be developed early in the project, concurrently with the project requirements specification and product development. It includes planning for integration sequence, activities and responsibilities involved, resources required, schedulable plans, procedures to be followed, required tools and technical environment, and human skills. The developed strategy should be reviewed by developers, test and integration teams for feasibility. It is necessary to revise continuously and different versions should be documented. As described in [19], the implementation of product integration processes depend on different aspects of software engineering such as influence of project life cycle models, interface design and management of interfaces and
organization of different kinds of projects (distributed development, component based development etc.).

Product integration focuses on collaboration between all the stakeholders in the product development [21]. This includes various efforts related to project lifecycle phases for a common view of the project and customer satisfaction. In [22], Martin Fowler proposes the collaboration between different stakeholders through wikis. Wikis are an effective mechanism to document and share the information related to design deadlines, build instructions, notes on the progress etc. It is also possible to provide feedback. When the iterations are small and a continuous integration occurs, it allows the customer to monitor the progress and identify the problems in time.

5.3 Problems in ABB Product Integration and IPA:

From the data obtained through the case study, the problems identified in the product integration are related to

- Misinterpretation of interfaces by the developers
- Inefficient code implementations
- Missing data when handover from development to integration
- Insufficient test coverage in particular before integration phase
- Difficulty in ordering test activities for example, what should be done before integration and what should be done after integration
- Insufficient documentation for complete system functionality,
- Lack of formal collaborations and missing feedback mechanisms
- Late discovery of problems related to interface definitions or inconsistent implementations of a given interface during different phases

Solutions suggested: process tool is recommended to provide consistent information on interface definitions, design process iterations etc. Further, continuous feedback should be documented on interface changes and consequences from the project activities. It connects project participants from different phases to work on consistent project objectives using the consistent shared information. As informal exchange of information has less impact on human minds (responsibility), the information should be available formally. In place of ad-hoc practices, a formal process should be defined and documented using a list of work products, roles, activities and guidance. The role specific processes should be defined to promote the process style of development. To achieve this, an efficient tool support is required. Adapting to a tool not only supports the product quality but also helps to improve the existing process.

As described above, to address specific product integration problems, an efficient integration strategy should be defined and it should be continuously revised based on the feedback from the project participants. To define an integration process, a relevant process tool should be used. As presented in previous chapters, Iris Process Author supports organizations for defining processes for all kinds of projects and provides an efficient feedback facility. The features of the tool are discussed in detail in
chapter 4. However, a specific tool’s applicability also depends on project characteristics such as project’s size, and its life cycle phases. There exist some statistical data in support of choosing a suitable tool for a given project with respect to its size, technologies, resource constraints etc [18]. For the two evaluated tools, though IPA is methodology independent, it supports project specific processes as well as the industry standard methodologies. By mapping the organizational processes and related structures with the standard frameworks such as CMM, COBIT, ISO etc, compliance reports can be generated to help organizations improve its processes.
Chapter 6
Conclusions and Discussion

For software development products, the key factors are quality, customer satisfaction, reduce development costs, and shorter development time. This demands continuous improvement of the development processes and effective use of related tools. An appropriate usage of a process tool in large software organizations, across multiple projects, ensures consistency and promotes the common understanding about the processes among development teams. In this thesis, we considered two process tools, namely IRIS Process Author (IPA) and Eclipse Process framework Composer (EPC), to evaluate the technical support to model the existing software development processes at ABB. To capture the existing process, an interview study was made. Based on process modeling and general aspects of a tool, an evaluation framework is considered to evaluate the tools. The tool-support for process authoring, tailoring, collaboration and reuse was investigated and specific comparisons were made. In specific to the organizational product integration processes, the involved problems were noted and also suggestions made on possible process improvements.

As the result of the tool evaluation, IPA is suggested for its suitability because of its support for collaboration and customization. Its repository feature called Iris Process Central (IPC), enables feedback management and process evolution. It is important that the previous experiences and lessons learned by the development teams are collected and documented in order to improve the existing processes. IPC provides different versions of processes with a variety of search options. Project management can locate the suitable processes for enactment. IPA has built-in support for import and customization of a wide range of industry methodologies. Also, it is observed that IPA has intuitive interface and user friendliness compared to EPC. Importantly, Automatic validation feature in IPA works at several levels of process authoring to ensure process integrity. But, as the tool is completely web based (and it can only support internet explorer as its default browser), it is important to enquire the offered private policy by the tool vendor and how the organizational information is allocated.

6.1. Limitations and Future work

The questionnaire for the case study was developed based on previous experiences in related field and the experts. However, this may deviate from the specific and critical aspects of the case study. The tool evaluation was mainly based on the data obtained through interviews. There was no possibility for involvement in the actual development process, which in turn could contribute to better evaluation framework. Also actual process details such as detailed activity steps, work products, guidance were not completely available.

From the case study, it is clear that relevant research results are needed for specific domains. For example, the integration phase of software development needs research efforts. The considered tools can be further investigated by extending evaluation framework with clearly available domain specific details.
References


## Appendix A

This appendix contains the template of the questionnaire used in the interview case study.

### Questionnaire on Software Integration Process at ABB

<table>
<thead>
<tr>
<th>Serial No.</th>
<th>Interviewee</th>
<th>Type</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>Organizational</td>
<td>How long have you been working for the company?</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>What is your previous experience? (what kind?)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>What are the characteristics of the projects you are currently working on? (ex: development, maintenance, research)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>a.) What is the duration of the current projects?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b.) Is it distributed among different countries or in-house project?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>c.) What kind of process do you follow in product development?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>d.) What is your role? (details)</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td></td>
<td>Project planning (details)</td>
<td>What is the project development process?</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td></td>
<td>Project Architecture and design</td>
<td>Could you please explain the architecture design process?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>a.) What are the key considerations taken into account while defining interfaces? (ex: tech leads..)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>b.) How do you handle refactoring when there is change in requirements?</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Handover from development to integration</td>
<td>Does anyone checks the quality of the artifacts produced by developer? (ex: code review)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>a.) How and when is it carried out?</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Integration</td>
<td>Can you describe the software integration process?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a.) What are the different stages in software integration process?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b.) How does integrators manage when the modules are not delivered on time?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c.) How do u trace the problems like build is not successful on integration due to incomplete updating of the repository?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>d.) what are the responsibilities of a developer at this stage?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>e.) How the hardware is integrated?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>f.) Integrated build is executed manually or automatically by a tool?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Testing</td>
<td>Can you describe your testing process?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a.) Do you feel it is required to improve test process at any instances (where and why)? if answer is no...How do u plan to improve it?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>b.) What is the time required between system testing and release date?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>General</td>
<td>Describe process of educating the team with company design and coding standards both at the beginning of the project and also during development if any changes occur(obvious in iterative process).</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a.) Describe the communication among the technical leads to fulfill their responsibilities.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>c.) What are the problems in software integration?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>d.) Any suggestions to improve integration process?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>e.) How would you benefit by having a process management tool?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix B

This appendix contains the snapshots of process authoring and published process using IPA.

User Profile in IPA:
View of development phases in IPA:

Defining precondition and goal for a phase:
Defining Iterations:

Updating a process element(activity):
State machine diagram for a work product:

Work breakdown elements:
Creating new guidance kind:

Work flow diagram:
Permitted library packages for user: A library package imports two available library packages

Delta report:
Work product dependency diagram:

Editing work product information from the diagram editor:
Work flow diagram – published:

Published wiki pages and actions can be performed:
Appendix C.

This appendix includes snap shots of process authoring and published process using EPF Composer:

Creating ABB-MDH Delivery Process Using abb.mdh configuration:
Process Authoring – Work Break Down Structure:

Publish Selected Process Wizard – abb_mdh_deliveryprocess is checked:
selecting publishing option to customize look and feel of the process
Published HTML page – snapshots from work breakdown structure

Published process element(role):

- **Role**: Base software developer
  - Contributes to the development of the base software platform.
  - Works in a manner to prevent usage of any base software that has suspected or shown faulty behavior in its intended environment.

**Relationships**

- Contribution to the development of the base software platform.
- Works in a manner to prevent usage of any base software that has suspected or shown faulty behavior in its intended environment.

**Additionally Performs**

- Conduct feature test
- Conduct subsystem test
- Conduct subsystem build

**Identifies**

- Base software