Software Process Notations

The Role and Quality of Described Processes

Per Johansson

DISSERTATION

International Project Management
CHALMERS UNIVERSITY OF TECHNOLOGY
NORTHUMBRIA UNIVERSITY
Göteborg, Sweden 2007

Dissertation for the degrees

MSc in International Project Management
Teknologie magisterexamen i International Project Management

Software Process Notations
The Role and Quality of Described Processes

Per Johansson

International Project Management
CHALMERS UNIVERSITY OF TECHNOLOGY
NORTHUMBRIA UNIVERSITY
Göteborg, Sweden 2007

"If you can't describe what you are doing as a process, you don't know what you're doing."

W. Edwards Deming (American Consultant, Statistician and Educator, 1900-1993)

Acknowledgements

This master’s thesis that forms my dissertation is the final step in my academic studies at Chalmers University of Technology and Northumbria University. The work that this dissertation is based upon was carried out at Ericsson AB during spring of 2007. There are several people that I wish to express my appreciation to and who are a part in that this dissertation exits.

First, I would like to thank my supervisor at Ericsson AB, Dr. Anna Börjesson and Carl-Magnus Olsson at the IT-University, Gothenburg for their support, suggestions and directions to keep me on the right track during all moments forming this long winding process writing a master´s thesis.

Secondly, I would like to thank Bengt Jansson at Gothenburg University for feedback and I will also thank all participants taking part in the interviews at Ericsson AB.

Last, but not least, I would like to express my gratefulness to my lovely wife Elin, for your love, support, and encouragement during the period when I was occupied with this master´s thesis.

Per Johansson Gothenburg, in April 2007

Table of Contents

Table of Figures

Software Process Notations
The Role and Quality of Described Processes

Per Johansson

Abstract. In today's competitive market for software developing organisations it is of high importance that businesses are effective in their development projects in order to deliver products with high quality at the right time and at a low cost. One way to enable this effectiveness is to describe software development processes in a coherent way to increase understanding between different target groups affected by the process. Ellmer and Merkl (1996) call the described development process for "organisational memory" and argue that this is highly important for successful software development.

In this study, we look at what process notations to use that can facilitate the described process. The study sets out by conducting a literature review of described software processes and notations. Specifically, this is done by reviewing two main questions: (1) what role the described software process serves and (2) what quality attributes are important to the specific organisation in the described software process. With that understanding in mind, we proceeded by reviewing the use of tools for modelling software processes. Based on the reviewed literature, a questionnaire withholding statements about the two questions was created. The empirical data for the study was collected in face to face interviewees with 12 employees working at senior management and senior engineer level at Ericsson (Lindholmen) in Sweden. At the interviews, the questionnaires were used to gather quantitative data and to get more depth to the study - qualitative data was also gathered during interviews. The result from our study reveals that main reasons identified for the role of the described process are in coherence with the reviewed literature. 'Storing organisational knowledge', 'discussing improvements' and 'communicating knowledge and competence' are features identified in the literature as well as during the interviews. Furthermore, analyses from our literature study conclude that there is no obvious common standard for process notations and process modelling. This was confirmed in our interview study as more or less none of the interviews had the same view on what process notation to use. Finally, the result from our study also reveals that main attributes for reaching high quality in the described process differs between what we found in the literature study and in the interview study. Literature promotes the importance of clearly defined roles, which was not promoted by our interviewees. On the contrary, our study indicates a need for presenting clear and understandable deliverables in the described software processes, which is not promoted in the literature.

Keywords: Software process, process modelling, process descriptions, process notations

1 Introduction

For an organisation, the process how to develop software and products is of high importance in today competitive market. Software process modelling tries to capture the main characteristics of the set of activities performed to obtain a software product (Acuna and Ferré, 2006). If the software processes are well documented and distributed among the organisation (i.e. in form of a process model), it is possible to reuse the described processes from earlier successfully projects into future projects and by this improving both the efficiency and productivity in order to achieve lower production costs. Ellmer and Merkl (1996, p. 60) argue that "a process model is an explicit representation of process knowledge and may thus serve as a means for storing and retrieving organisational knowledge about software process execution". While software development today makes considerable use of varying degrees of formalized described software processes to give guidance to their work, the increasing interest in distributed and global software development has brought the matter of relying on more control oriented processes.

As an effect of the distributed development interests, the question of how to effectively describe the software processes becomes equally important. Somewhat surprising, while both control oriented approaches such as, Capability Maturity Model (CMM) and ISO 9000 (Paulk et al. 1993; Mcmanus and Wood-Harper, 2003) and Agile approaches such as, Scrum and XP (Abrahamson et al. 2003; Nerur et al. 2005) describe ways of working, there is little to no help for organisations from either approach in selecting the appropriate process notation. Inevitably, this is likely to be one of the reasons why arbitrarily defined bubbles and arrows, using tools such as Microsoft Power Point or Microsoft Visio, has emerged as one of the most common ways to describe software processes. For instance, Davies et al. (2006) show that in process modelling two of the most frequent techniques used today are workflow modelling and Unified Modelling Language (UML) and that the most frequent tools used for this modelling are Microsoft Visio and Power Point. Disadvantages (Ellmer and Merkl, 1996) with these ad hoc descriptions are: (1) the creator of a software process model is the individual that knows how to interpret the bubbles and arrows and it might be hard for other users to understand the process, (2) the knowledge of a software process model might be hidden and other users can't find it and therefore it is not possible to reuse it, and thus may become an obstacle for sharing the organisational knowledge.

Commonly, documenting software processes is associated with management of personnel but should be seen as a way to help and support software developers in their work (Davies et al. 2006). Well functional described software processes are a base as presented above to increase efficiency as well as to increase product quality. To implement what Ellmer and Merkl (1996) call "organisational memory" they argue that there need to be a process model library implemented and mechanisms have to be provided for retrieving and tailoring process models. An example of a tool for keeping a model library is the portfolio management tool 'Rational Method Composer' which gathers and distributes the organisational knowledge within an organisation. With this tool it is possible for projects and project managers to reuse already proven methods and software processes. However, Iivari (1995) have studied the perceptions of effectiveness of tools for modelling processes and he found that these were considerably low.

Large organisations are generally good in documenting the developed product and modelling the software code is but the process how to develop the software could be better documented. Usually an organisation has one standard to document a process for each department which causing problems when resources move between departments. To share the described software processes an organisation needs common techniques, principles, and tools that support this work. According to Davies et al. (2006, pp. 361) comparatively little empirical work has been undertaken on modelling in practice. Both Floyd (1986) and Necco et al. (1987) have conducted empirical work on modelling techniques in practice but this work is now out of date (Davies et al. 2006). Responding to this, we review how organisations can reach an understanding of what software process notations to use. Specifically, this is done by assessing two main questions: (1) what role the described software process serves and (2) what quality attributes are important to the specific organization in the described software process. With that understanding in mind, we proceed by reviewing the use of tools for modelling software processes. Conducting this three folded review is not a straightforward task, as related research reveals an array of potential roles for describing software processes and an equally mixed range of important matters to achieve high quality in the described software processes. On top of this, the mapping of what quality attributes are important for specific roles that the described software process plays is far from clear. Thus, it becomes evident that a case study where a particular organization and its' viewpoints on the use and usefulness of described software processes is called for. Only once we have such understanding does it become feasible to elaborate on what tools may be appropriate to use for a particular organization. One suitable area to find organisations that operate largely on a global software development level is the telecom sector, and upon inquiries it became apparent that Ericsson was indeed highly interested in assessing the described software process work they have initiated. For Ericsson, the study represents a valuable opportunity to get a formal review of the main options for software process modelling, as their work in that area so far has been done using a tool from a supplier based on an existing business deal rather than a strategic decision.

Our approach to this research was to make an extensive literature survey (see chapter four for how the survey was carried out) for different process modelling techniques and notations. We found that the existing business process modelling techniques are dominated by three different techniques; activity process oriented techniques, object-oriented techniques and role oriented techniques, these will be presented in more detail at the literature review chapter. Knowledge about the role of described software processes and the attributes leading to high process quality was also obtained during the literature review. From this review, we created questions and gathered empirical data of senior managements and senior engineers view on described software processes. The data was gathered in semi-structured interviews with the participants. All participants were chosen based on their will to participate and in order to map a good combination of managers, senior development engineers and process engineers.

The continuing two chapters that follow in this study withhold a literature review (i.e. the theoretical framework). A description about the method that is used in the study can be found in chapter four. In chapter five we present the result from interviews (empirical data) in form of quantitative data and qualitative data. With both the reviewed literature and the empirical data at hand a discussion is made in chapter six, where the theoretical framework is applied on the empirical data. Chapter six is also containing an implication to further research. This study ends with chapter seven summarising this work with our conclusions and contributions.

2 Literature Review: Dominating Process Notations

To be able to describe software processes, we need a good modelling technique that includes 'modelling tools' and notations. For this Kueng et al. (1996) proposes a taxonomy of business modelling techniques (used for describing business processes) and they classifies the techniques into four groups:

- Activity-oriented techniques. These techniques are focusing on the definition of business processes as a sequence of activities. Examples on Activity-oriented techniques are further discussed by Owen and Raj (2005), and White (2004).
- Object-oriented techniques. These techniques are leveraging the more comprehensive modelling constructs of object-orientation to capture business processes. Object-oriented techniques are further discussed by Kalnins and Vitolins (2006), Haumer (2006), and Russel et al. (1994).
- Role-oriented techniques. These techniques describes modelling business processes based on the specific organisational roles and responsibilities involved and these techniques are discussed by Ould & Huckvale (1995), and Ould (1995).
- Speech-act oriented techniques. The Speech-act oriented techniques are viewing business processes in the context of the speech-act language action paradigm. These techniques are more discussed by Kueng et al. (1996).

It is not obvious whether speech-act oriented techniques are primarily dedicated to analysing existing processes or for creating new processes (Kueng et al. 1996), and due to this unclearness there is few articles to be found about this technique. As a consequence, this study will focus on the first three techniques and leave the Speech-act approaches as a subject for further research to be done.

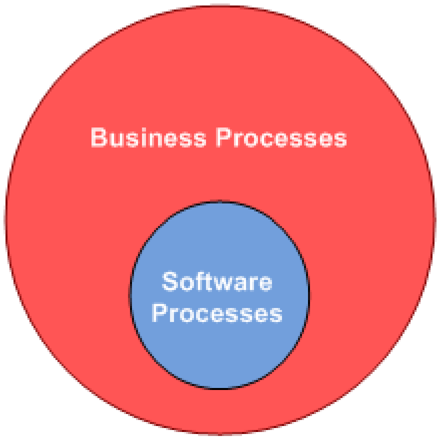


Figure 1. Software processes as subset

When we discuss business processes further on in this study we will assume that software development processes is a subset of business processes as shown in Figure 1. The software development processes will therefore in this study further on be referred to as software processes.

2.1 *Why Described Software Processes?*

In these days the described software process is a critical factor for businesses to deliver quality in software systems. A software process model can be seen as an abstract representation of the architecture, design or definition of the software process (Acuna and Ferré, 2006). In this study a software process is defined by the set of activities required to produce a software system. The definition of a software process usually specifies the actors executing the activities, their roles and the artefacts produced. This is defined by Finkelstein et al. (1994) who see a process model as the description of process expressed in a suitable process modelling language. Software process modelling tries to capture the main characteristics of the set of activities performed to obtain a software product. Different elements of a process can be activities, artefacts (products), resources (personnel and tools) and roles which all can be modelled. There several proposals of what kind of main elements that can be modelled (Acuna and Ferré, 2006). In the Software Process Engineering Meta-model (SPEM) standard (Haumer, 2006) the suggested elements in a process are role, task and work-product. Figure 2 show the elements and their interrelationships according to Haumer (2006). A role represents a set of skills, competences and responsibilities. In order to create or to modify work products a role is assigned to perform a task. The tasks can have both inputs and outputs in form of artefacts.

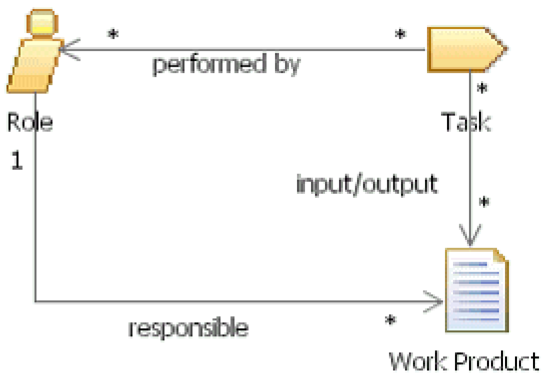


Figure 2. Interrelationships in a process

There are different kinds of information that people want to retrieve from a process model (Curtis et al. 1992, pp.77) such as what is going to be done, who is going to do it, when and where will it be done, how and why will it be done, and, who is dependent on its being done. The software processes can be described in a model so that it becomes clear in what order they should be performed and as presented above. If a described process model is used there are according to Ould and Huckvale (1995) several ways that this model can help organisations to be more effective. A described process model can be used in several roles;

- *As a focus for discussion*. If we use a good modelling technique that have a clear syntax and semantics together with a disciplined process (modelling guidelines) for creating the model, it will help the organisation to ask the right questions and bring important points in focus for discussion.
- *As a means for communicating a process to others*. In this way other people can use the process as a guide. In this way they will save valuable time since they don't have to develop a new process over and over again. The means of communicating is also mentioned by Curtis et al. (1992) where they also add "ease of understanding" as a means to other users.
- *As a basis for analysis*. It is possible by analysing the model for users to find possible weaknesses in the processes or to identify if certain processes don't give any value and therefore can be deleted or replaced.
- *For designing a new process*. When creating different process models that are based over the same process and comparing these it is possible to find the best solution and to use this.
- *As a baseline for continuing process improvement*. It is possible that the process models can be used to find different measures in how well the process works and in this way start working with improvements to the process. This can be used as a program for controlling the 'real world' process. If the process model is formal enough it can be used in a workflow management system and later be executed in a

computer system. This view 'process improvement support' is also supported by Curtis et al. (1992).

Additional goals and benefits of modelling the software process (Curtis et al. 1992, pp.77) are; *Support process management* that requires a project-specific software process and monitoring, management and co-ordination. *Automated guidance in performing process,* this requires a definition of effective software development environment, providing user orientations, instructions and reference material, retain reusable process representation in repository. *Provision for automated execution support:* requiring "automated process parts, support co-operative work, automatically collect measurement data reflecting actual experience with a process".

For the benefits of reusing software process models in a repository mentioned by Curtis et al. (1992) as a sort of "organisational learning" Ellmer and Merkl (1996) gives several arguments and a couple of them are; (1) It is possible that the knowledge gained during years of using the processes can be lost if for instance an engineer leaves and in storing the knowledge the risk losing it are reduced.. (2) The amount of time that can be saved by reusing already well functional processes are significant and in this way costs can be reduced and the productivity increased. (3) Since the processes that are reused already are tested it will increase the quality of the products since the quality of the development process is increased. (4) When reusing parts of the software process this might be an important step towards implementing and using a software process improvement standard such as Capability Maturity Model (CMM).

A common problem in organisations today is that there are discrepancies between the actual behaviour (daily work) and a stated process in an organisation. Reasons for this discrepancy might be high-level prescriptive processes that are unrelated to actual project activities, imprecise, ambiguous, incomprehensible or unusable descriptions of processes to be performed, and there might be a failure to update the documentation as the processes changes (Curtis et al. 1992).

There are as presented above several reasons for organisations or business to use process modelling (Davies et al. 2006; Ellmer and Merkl (1996) in their work of documenting and describing all software processes. Katzenstein and Lerch (2000) claims that previous research shows that the representation of a problem in form of a modelling language can affect the quality at the solutions created. It is therefore very important to be able to judge the quality at the models that are created, not least models of processes.

2.2 Three Main Modelling Techniques

The following text in this section will describe the three modelling techniques presented above. The first technique to be described is the Activity-oriented technique this is followed by the Object-oriented technique and the section ends with a description of Role-oriented techniques. In each section there will also be examples on a notation for each modelling technique presented. The software process that is presented in the examples is describing the development of product (including both software and hardware) Alfa at Ericsson.

2.2.1 Activity-oriented Technique

Business Process Modelling Notation, BPMN is a notation that can be used in Activity-oriented techniques. BPMN is a notation that is issued from Business Project Management Initiative (BPMI, 2007) and the BPMN standards can be found at BPMN (2007).When using BPMN a Business Process Diagram (BPD) is constructed and this diagram is based on a flow charting technique that is tailored for creating graphical models of business process operations. The BPMN is designed to be easy to use and easy to understand, but it also was created to provide the ability to model complex business processes. Further BPMN is also designed having webServices (Wikipedia, 2007) in mind and BPMN is the standard notation for how to develop webServices that are based on business processes. The procedure is that processes are described with BPMN that produces as BPD and this is then published in a Business Process Execution Language, BPEL (Owen and Raj, 2005). There are however competing standards proposing a BPEL. Large corporations such as Microsoft, IBM, and BEA have with a joint venture together created an industry standard named Business Process Execution Language for webServices (BPEL4WS). It is also possible for BPMN to be used only as a notation describing a process and it don't have to be executed in an execution language.

There are several Computer-Aided Software Engineering tools (CASE-tools) that support BPMN and this is due to that it can be used to model webServices and deliver them running on BPEL and BPEL4WS platforms. The largest vendors are Borland (product: Together), BEA (product: Fuego5), IDS-Sheer (product: Aris), IBM (product: WBI Modeler), Telelogic (product: System Architect) and Proforma (product: ProVison). More CASE-tools can be found at http://www.bpmn.org/BPMN_Supporters.htm.

2.2.2 Notation and Principles for BPMN

BPMN has been designed for doing two things well (Owen and Raj, 2005) - BPMN should be easy to use and it should be understandable for non-technical users (usually management). BPMN was designed in order to offer expressiveness for modelling highly complex business processes and to be mapped against business execution languages (Owen and Raj, 2005). The main principle in BPMN is to use a single business process diagram and in BPMN there are four categories of elements, these are according to Wang et al. (2006).

-
- Flow objects
  - Event
  - Activity
  - Gateway
- Swim lanes
  - Pool
  - Lane
- Connecting objects
  - Sequence flow
  - Message flow
  - Association
- Artifacts
  - Data object
  - Group
  - Annotation

In BPMN the concept of "sequence flow" is used. This means that a process is described in" a flow" where we model the events that occur to start a process, the process that get performed, and the results of the process flow. A gateway is then used to control the divergence and convergence of the sequence flow in the diagram. A process in the flow can contain a sub-process and if so this is marked with a + character as an indication of this. This sub-process can be shown by another BPD (the sub-processes in BPD is connected via a hyperlink). The lowest level of a process in BPMN is called a task and a task cannot be de-composed (Owen and Raj, 2005). If desirable there are group and annotation which are used for documentation to make the BPD better understandable and they do not affect the flow of the process.

According to Owen and Raj (2005) the BPMN have advantages over UML as a notation for modelling processes because that BPMN offers a process flow modelling technique that is more conductive to the way business analysts model. Secondly BPMN has mathematical foundation that makes it possibly to map it to a BPEL. In UML this is not possible to do according Owen and Raj (2005).

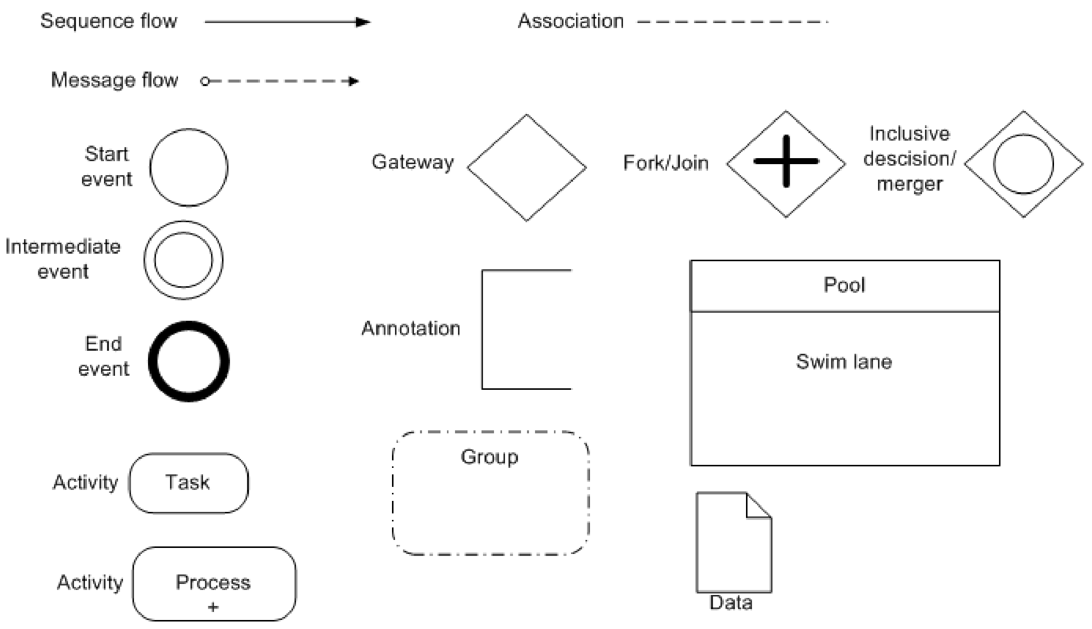


Figure 3. Basic elements of BPMN

In Figure 3 the main elements of BPMN is briefly presented and for a more comprehensive description we refer to Appendix A where a description of each of the elements in Figure 3 is presented. For a more comprehensive description for all available elements we refer to Owen and Raj (2005) or to BPMN (2007).

2.2.3 Example on a Process Model in Business Process Modelling Notation

Presented below in this section is a software process described using BPMN, the same software process will later be described in a UML 2.0 Activity Diagram and partly in Role Activity Diagram. The process is adapted from the RMC and shows the development of the product Alfa at Ericsson. In the software process model there are four levels in describing the development which are used where Figure 4 shows the first level, Figure 5 shows the second level, and Figure 6 shows the third level. Figure 5 is the sub-process for ‘System Analysis and Design’ in Figure 4 and Figure 5 is the sub-process for ‘Analyse the System’ in Figure 5. The following abbreviations are used in the figures presented below CPI- Customer Product Information, FB - Functional Block, RBS – Radio Base System, SW- Software, and HW-Hardware.

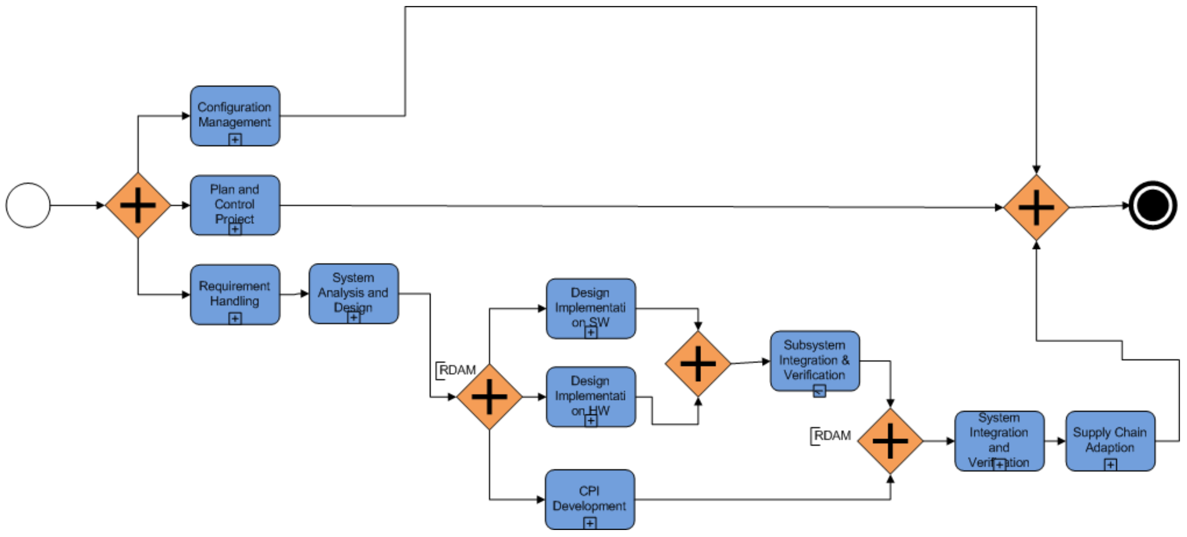


Figure 4. Software process in BPMN

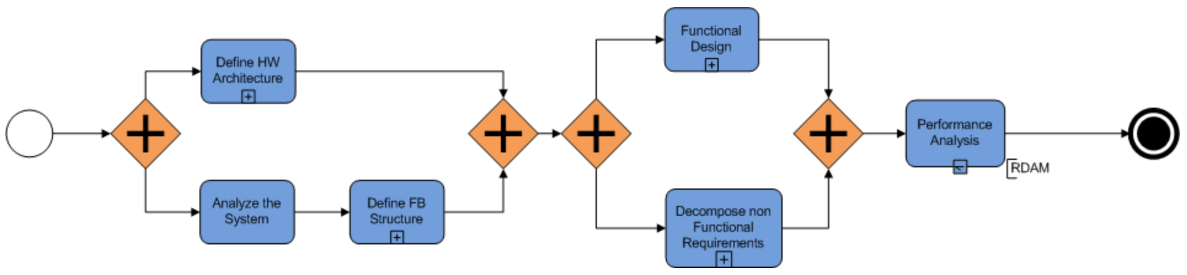


Figure 5. Sub process System Analysis and Design in BPMN

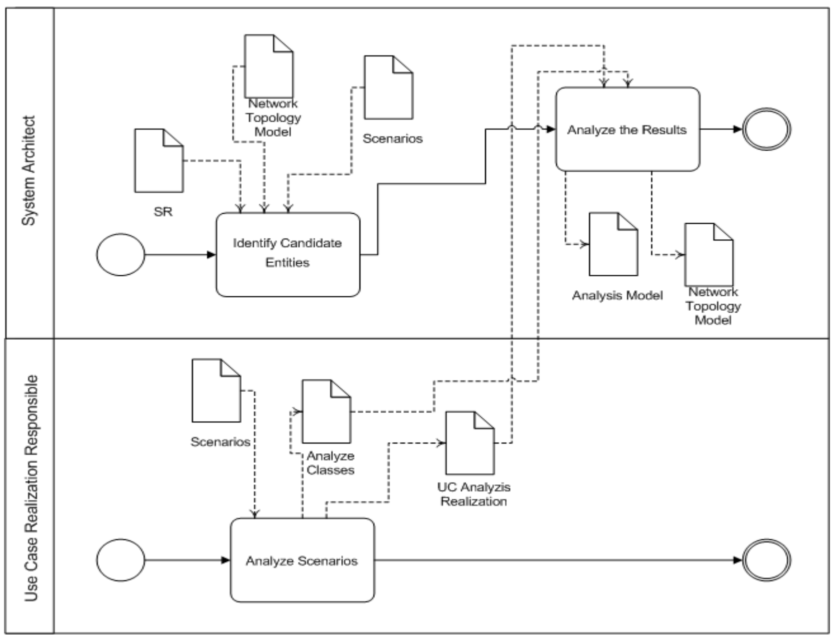


Figure 6. BPMN diagram based on roles.

2.2.4 Object Oriented Techniques

Unified Modelling Language (UML) is an example of an object-oriented modelling language. UML is a visual language for specifying, constructing and documenting artefacts of systems and UML is being seen as the de-facto standard for software modelling and design (Russel et al. 2006), (Tilley and Huang, 2003). UML is a language that is developed by OMG (OMG, 2007a) that is an international, open membership, not-for-profit computer industry consortium. UML is a general purpose language which means that UML can be used for modelling in different domains such as software, business processes and requirements. Generally UML is used when designing software developing projects using object oriented techniques. Today UML is in version 2.0 and the major releases of UML have been, UML 1.0, UML 1.3, UML 1.4 and UML 1.5.

There are several related notations included in UML, including Use Case, Class, Interaction, State and Physical Diagrams as well as Activity Diagrams. UML 2.0 consists of thirteen types of diagrams, divided into three categories: Six diagram types represent static application structure; three represent general types of behaviour; and four represent different aspects of interactions (Holt, 2004):

- Structure Diagrams include the class diagram, object diagram, component diagram, composite structure diagram, package diagram, and deployment diagram. Structure diagrams capture the underlying static structure of a software system at various levels from individual objects to overall application packages (Russel et al. 2006).
- Behaviour Diagrams include the use case diagram (used by some methodologies during requirements gathering). Activity Diagram and state machine diagram. Behaviour diagrams describe the overall functionality of the software at a relatively high level of abstraction (Russel et al. 2006).
- Interaction Diagrams, all derived from the more general behaviour diagram which include the sequence diagram, communication diagram, timing diagram, and interaction overview diagram. Interaction diagrams further augment the behaviour diagrams and present a more detailed description of system functionality in terms of object interactions (Russel et al. 2006).

UML 2.0 Activity Diagrams are the object oriented equivalent of flow charts and data-flow diagrams from structured development. In UML 1.X, UML Activity Diagrams were a specialization of UML state machine diagrams but in UML 2.0 those have been replaced with activity invocations (Holt, 2004). Activity Diagrams are not an original part of UML and they were added later, and draw on a number of earlier techniques for modelling events, states, and workflows (Odeh et al. 2006). UML 2.0 Activity Diagrams can be used to explore the logic of (Ambler, 2005) a complex operation, a complex business rule, a single use case, several use cases, a business process, concurrent processes, and software processes. Activity Diagrams appear to be an attempt to project or extrapolate an object-oriented approach into the organizational context of the software development process (Odeh et al. 2006). UML 2.0 Activity Diagram describes sequencing of activities and they supports conditional and parallel behaviour. This is useful for analysis of workflow and parallel processes. Holt (2004) claims that there are two main uses for UML 2.0 Activity Diagrams which are to model workflows and operations, mostly UML 2.0 Activity Diagrams are used to model workflows. The term workflow can according to Holt (2004) be a little confusing since the word workflow is widely associated with the Rational Unified Process (RUP) and Holt gives two examples of how to use UML 2.0 Activity Diagrams, one interpretation according to RUP and one interpretation according to ISO. Examples on this can be seen in Figure 7 as the ISO-interpretation is used and in Figure 8 where the RUP-interpretation is used.

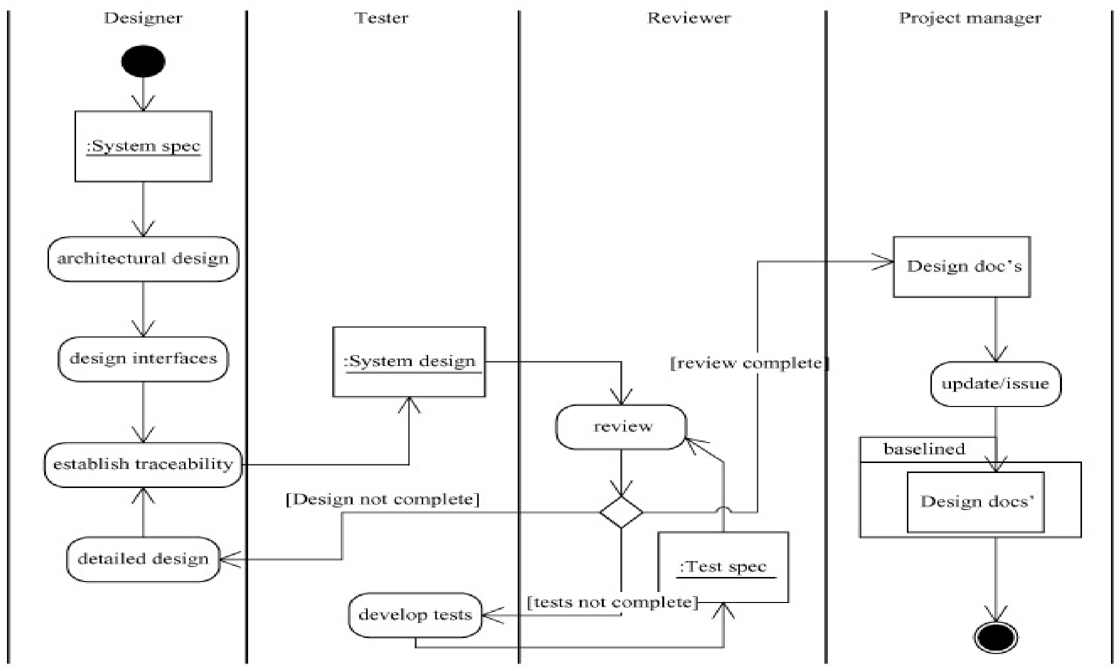


Figure 7. UML Activity Diagram according to ISO. Source: (Holt, 2004)

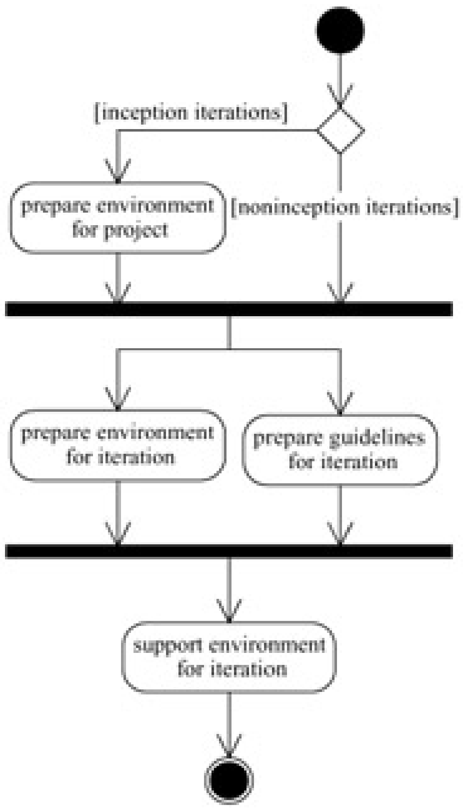

[inception iterations]
prepare environment for project
[noninception iterations]
prepare environment for iteration
prepare guidelines for iteration
support environment for iteration

Figure 8. UML Activity Diagram according to RUP. Source: (Holt, 2004)

It is the UML 2.0 Activity Diagram according to RUP that should be used to model workflows (Holt, 2004).

There are several arguments in favour for to use UML as a modelling language for software processes and what to think about when choosing a notation for software process modelling. First point is that UML have robustness, it provides functionality for a graphical representation of various software processes and needs. Secondly UML is a flexible notation because it can be used on all stages of a software process improvement project. Third point is that UML is widely recognised and that it provides support for software process analysis as well as technical support (Lyalin and Williams, 2006). UML also provides a large set of diagrams that can be used to define both structure and behaviour of dynamic software processes (Jäger et al. 1999).

There are a high number of vendors for CASE- tools that support software process modelling with UML. The largest vendors are IBM (product: Rational Software Architect and Modeller), Borland (product: Together), Telelogics (products: TauDeveloper and TauArchitect) and Altova (product: UModel - UML 2.1). More CASE-tools supporting UML can be found at OMG (2007b). What to take notion about is that most of the presented CASE-tools don't have support for modelling with UML 2.0 Activity Diagram since Activity Diagram is the least used of all UMLs diagrams (Holt, 2004).

2.2.5 Notation and Principles for UML

In Figure 9 the graphical representation of the UML 2.0 Activity Diagram most common elements can be seen. At Appendix B a description of the symbols can be found. If a more comprehensive description of the elements is preferred, study (Holt, 2004) or (Ambler, 2005).

It is possible to show partitions in two ways, either in swim-lanes or by writing the partition name in brackets above the activity name in the activity invocation symbol, see Figure 7 and Figure 8 for example on this.

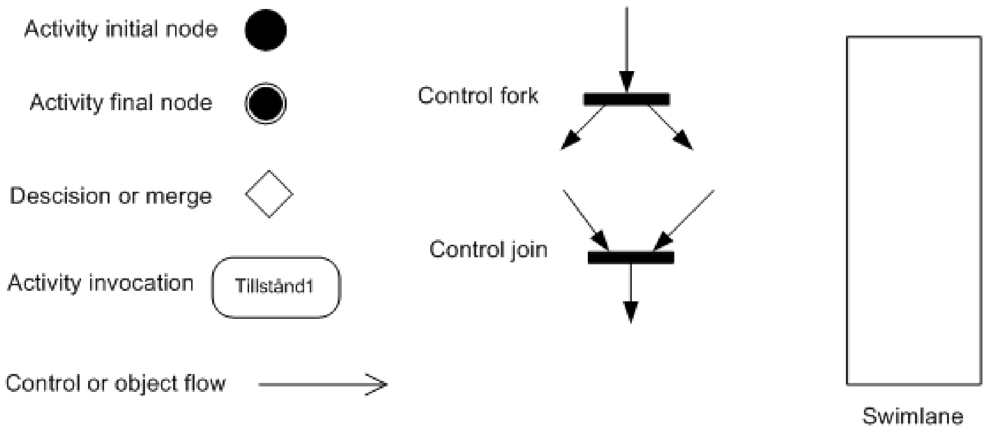


Figure 9. Elements of UML 2.0 Activity Diagram notation.

In Appendix B there is a brief explanation of each of the different elements based on Figure 9.

Software Process Engineering Meta-model (SPEM) is a current OMG industry standard that was originally developed by IBM and given to OMG. SPEM is process engineering meta-model as well as a conceptual framework. SPEM supports concepts for modelling, documenting, presenting, managing, interchanging and enacting development methods and processes (Haumer, 2006). The most current release of SPEM is version 2.0.

The idea of SPEM is to separate reusable method content from the described software processes which can be seen in Figure 10. Because of this the processes that are contained are reusable i.e. role, artefacts, tasks etc. The word "artefacts" has in SPEM version 2.0 been changed to "Work Product".

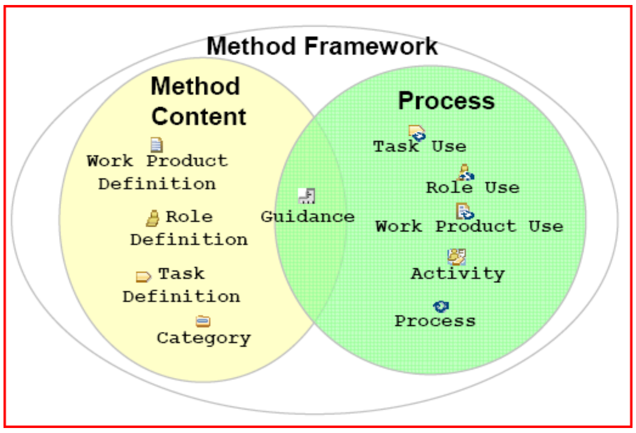


Figure 10 . Method framework. Source: (Haumer, 2006)

As an example SPEM 2.0 make it possible to reuse described software processes or described software process patterns and tailoring those described software processes allowing users or projects to define their own extensions and variability. This is helpful for an organisation using CMM (Haumer, 2006). Several described software processes can refer to the same method content and changes made in the method will be reflected in all described software processes that using it. For instance it is possible to share described software processes between different methodologies such as RUP and Agile methods (Haumer, 2006). The Guidance area is between method content and described software process and is additional information such as templates, checklists and tool mentors.

The Unified Method Architecture (UMA) is developed by IBM and its intention is that it shall be used to model software processes and be used to develop tools that support software process engineering, e.g. RMC (Software from IBM for portfolio management). UMA is an evolution of the Software Process Engineering Meta-model (SPEM). According to Haumer (2006, p. 23) it should be possible to use both UML 2.0 and BPMN to describe an actual development software process and then execute the software process in a BPEL-based workflow engine. In RMC it is possible to use plug-ins such as an Agile plug-in that makes it possible to use RMC in a company using agile methods in development of software.

UMA (SPEM 2.0) consist of three different diagrams based on UML 2.0-diagrams. A Workflow Diagram that represents a flow of activities. Activity Detail Diagram that includes the element tasks, role and work product. This diagram has it focus in the details of an activity, what outputs the tasks produced, the input that the task requires and the role that performs the work. The third diagram is the Work Product Diagram and this diagram shows the work products dependencies.

2.2.6 Example on a Software Process in UML 2.0 Activity Diagram

Presented below is a model of a software process how to develop the Ericsson product Alfa described by using UML 2.0 Activity Diagram (RUP-interpretation). The same process is described earlier in the section when discussing the same software processes described in BPMN. The same abbreviations as used in BPMN earlier presented are also used in the diagrams presented in Figure 11 and Figure 12.

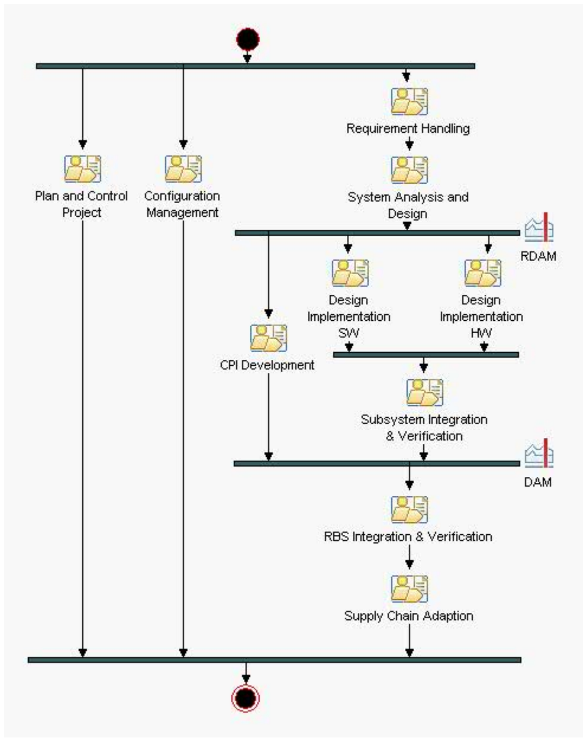


Figure 11. Software process in UML

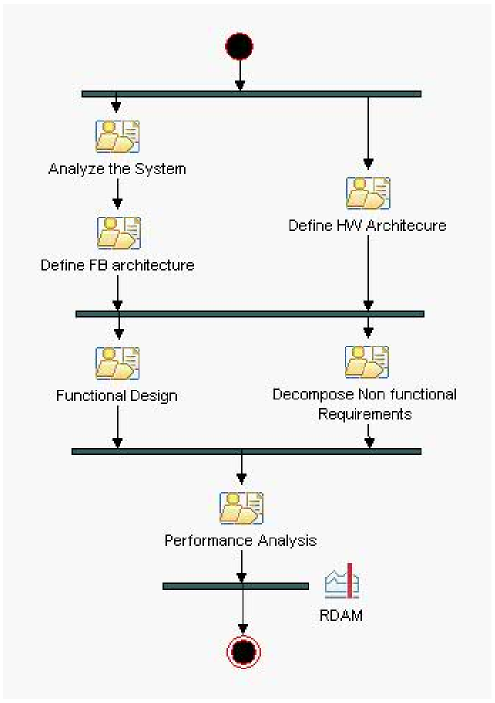


Figure 12. The sub process System Analysis and Design in UML Activity Diagram

Both the Activity Diagrams presented in Figure 11 and Figure 12 has the SPEM 2.0 notation but the logic is the same as in the original UML 2.0 Activity Diagram. The diagram in Figure 13 is decomposed from "System Analysis and Design" in Figure 12. Figure 13 is decomposed from "Analyze the System" in Figure 12 and this shows the tasks in the activities and what the roles associated to this task should produce in form of artefacts. The artefacts can be both input and output from the tasks.

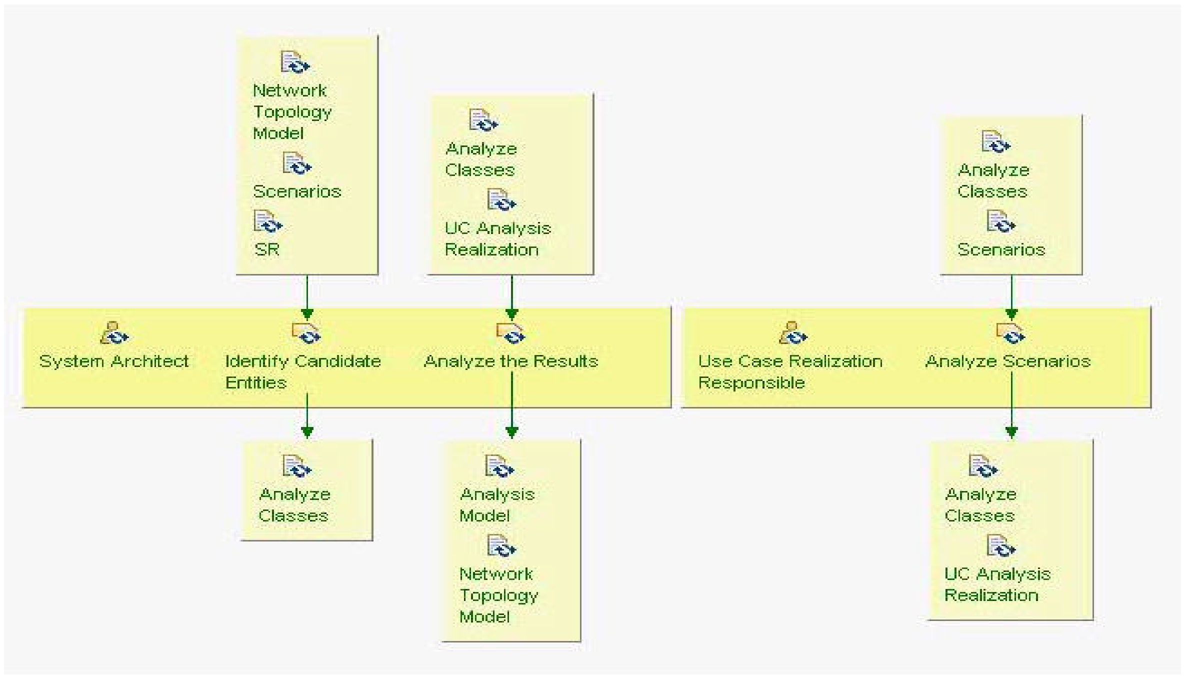


Figure 13. Roles, tasks and artifacts according to RMC.

2.2.7 Role-oriented Technique

Role Activity Diagramming (RAD) is developed as a notation for describing different kinds of processes. The language has been developed by Praxis Inc. and it is included in a modelling technique that is called Systematic Technique for Role and Interaction Modelling (STRIM). STRIM was originally developed by Praxis Inc. to support business process reengineering.

Examples in how to use diagrams created in RAD for modelling software processes can be found in Murdoch and McDermid (2000) and Nandish (2000). Nandish shows that RAD is well suited as a "socio-technical" method to be used in the National Health Care in United Kingdom (NHS) and that it brings forth the important roles in a process and the interaction and collaboration required achieving the goals of the process. Murdoch and McDermid (2000, p.64) concludes in their article that "the RAD and Role Context Chart notations are readily understandable by engineers from a range of disciplines and provide a means of linking business, management, and engineering processes".

2.2.8 Notation and Principles for RAD

In Figure 14 the graphical representation of the Role Activity Diagrams most common elements can be seen. In Appendix C a description of the symbols can be found.

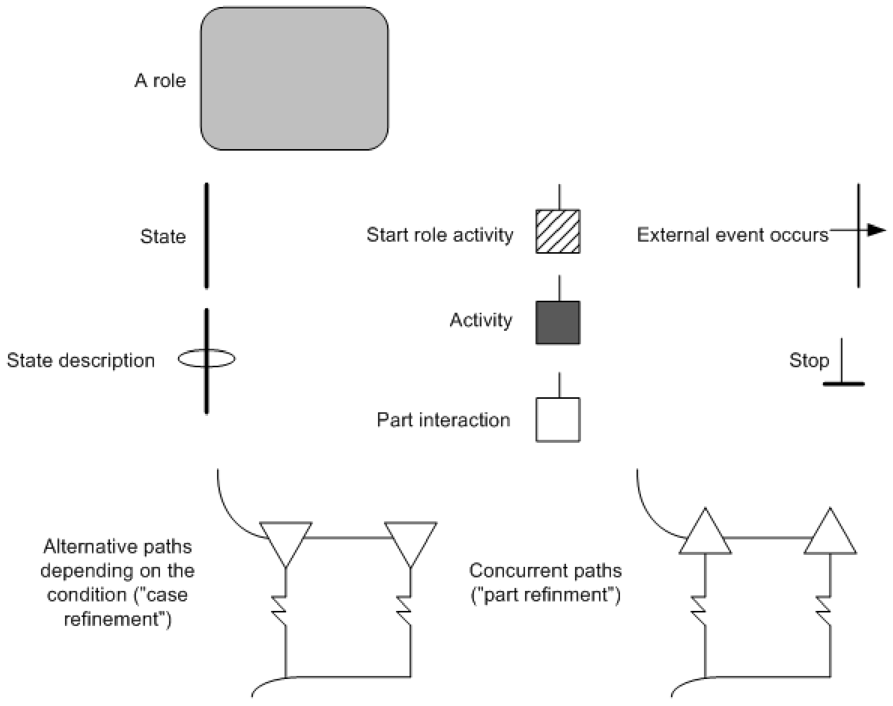

A role
State
Start role activity
External event occurs
Activity
State description
Stop
Part interaction
Alternative paths depending on the condition ("case refinement")
Concurrent paths ("part refinment")

Figure 14. Main elements in RAD notation.

Role Activity Diagramming is described by Ould (1995) and the RAD notation according to Ould is summarized in Figure 14. A more comprehensive information how and when to use RAD, can be found at Ould and Huckvale (1995) and Ould (1995).

RAD is used to model what will perform certain activities in a process but also what the resources do and how they do it and according to Ould and Huckvale (1995, p.337) there are five key concepts that have to be modelled for a business processes in STRIM and those are:

1) How activities are divided amongst roles
2) What the organisation is trying to achieve with the process: the process goals.
3) What people do to achieve the goals: activities
4) How people within groups interact collaboratively to get the job done
5) What constraints the organisation puts on what people can do and how they should operate: the business rules.

In Appendix C there is a description of the RAD notation and the symbols used in this notation (Ould and Huckvale, 1995).

2.2.9 Example on a Process Model in Role Activity Diagramming

The example below is the same described software process that has been modelled in UML and BPMN in earlier examples but now it is modelled in Role Activity Diagramming. It shows the described software process developing the Ericsson product Alfa. Figure 14 is the sub-process of 'System Analysis and Design' in Figure 15 and Figure 17 is the sub-process of 'Analyse the System' in Figure 16. The same abbreviations as used earlier in the BPMN section are also used in this model.

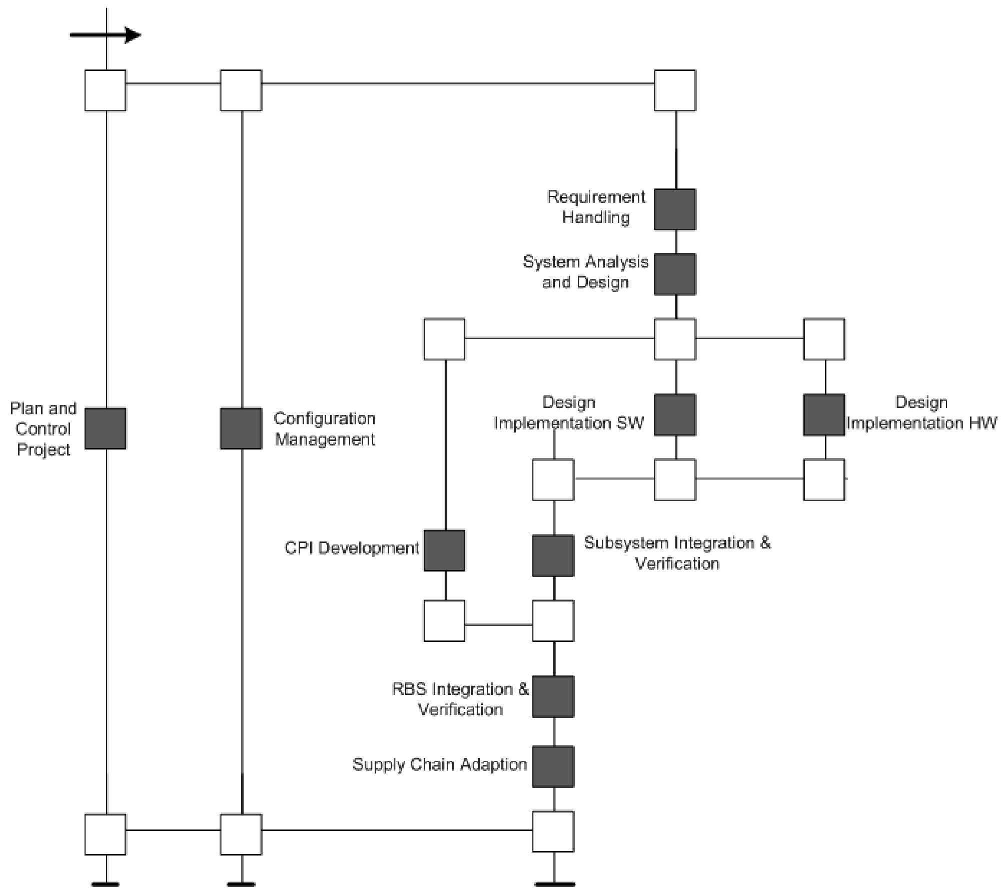


Figure 15. ALFA overall process in RAD

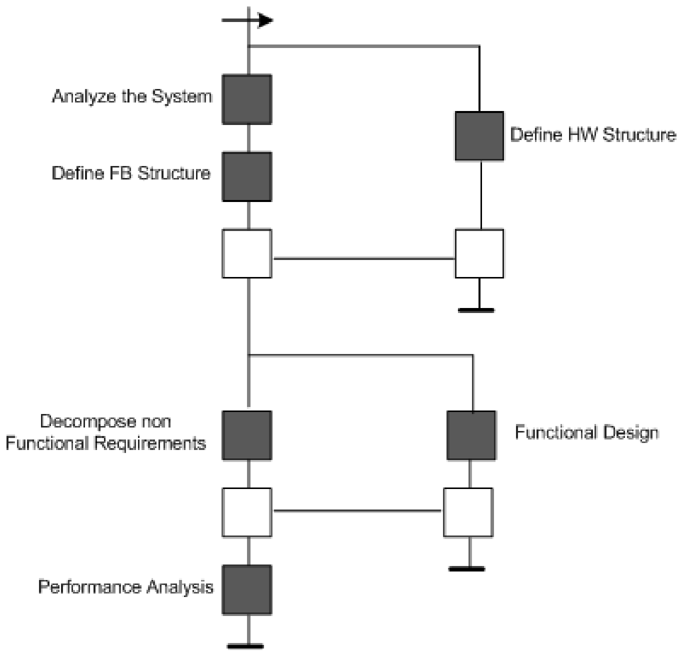


Figure 16. The sub process "System Analysis and Design" in RAD.

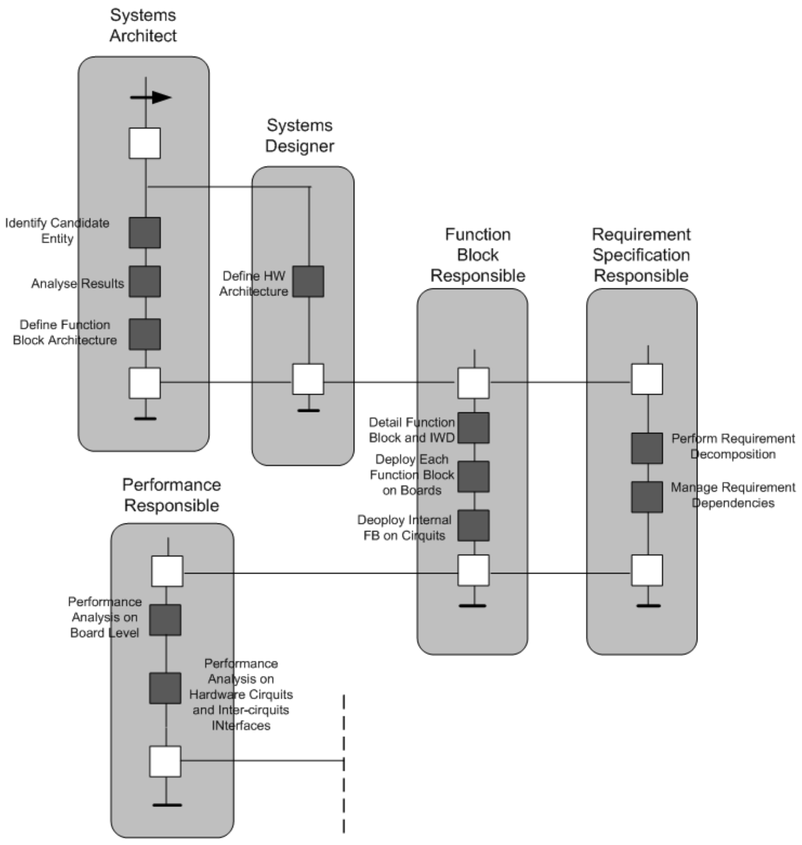

Systems Architect
Identify Candidate Entity
Analyse Results
Define Function Block Architecture
Systems Designer
Define HW Architecture
Function Block Responsible
Detail Function Block and IWD
Deploy Each Function Block on Boards
Deoploy Internal FB on Cirquits
Requirement Specification Responsible
Perform Requirement Decomposition
Manage Requirement Dependencies
Performance Responsible
Performance Analysis on Board Level
Performance Analysis on Hardware Cirquits and Inter-cirquits INterfaces

Figure 17. Role Activity Diagram

The example in Figure 17 is the same process that is described in Figure 6 and Figure 13.

3 Assessing Described Software Processes

As looking back to the two main research questions, we are now faced with the task of making sense out of the three dominating process notations reported on in the literature review (section 2) in relation to the role and quality of described software processes. Presented below is thus an outline of the relevant research that later will serve as foundation to the questions directed at our 12 respondents during the interviews.

3.1 *Quality in Described Processes*

Some of the benefits from modelling software processes in a large software developing organization are that parts of the software development process can be reused (Haumer, 2006) and thus time is saved at design phase when it is not necessary to invent the same things over and over again. This is an example on that the organisational knowledge can be shared according to Ellmer and Merkl (1996). In this way it is possible for projects to start quickly in that project managers can reuse already tested and refined parts/processes of earlier successfully projects.

There are according to Ould and Huckvale (1995, p.334) some features that must be met to obtain a high-quality process modelling notation:

- The concepts, objects and relationships represented in the model should be intuitively familiar, so that people can readily understand and talk about them.
- The notation should be easy for readers to grasp, after (say) a fifteen minutes verbal introduction.
- The notation should be unambiguous (another way of saying that it should have formal syntax and semantics) so that it can be analysed and, possibly enacted.
- It should be possible for the notation to draw attention to the purposes of what people do rather than the detail of how they do it; this requires a concept for relationships between people.
- The notation should be possible to handle complexity.

Most of the methods for business and software process models and notations are based on standards from independent non-profit organisations such as BPMI (Business Process Modelling Initiative), WfMC (Workflow Management Coalition) and OMG (Object Management Group). There has recently risen work for a new OMG initiative to join UML 2.0 Activity Diagram and BPMN under one integrated meta-model (Kalnins and Vitolins, 2006). This is due to that BPMI has become a part of OMG (Owen and Raj, 2005).

There are articles comparing different kinds of notations that state different positions of which of the standard notations that will become the common standard for business and described software processes. Kalnins and Vitolins (2006) claims in a study that UML 2.0 Activity Diagram are best suited as the standard as a notation for business processes whereas Wang et al. (2006) in another study claims that BPMN is the best suited notation for modelling business processes because UML 2.0 Activity Diagram is a heavy weight tool. A study (White, 2004) looked at 21 different workflow patterns in comparing BPMN and UML 2.0 Activity Diagram. The study found and concluded that both notations provide similar solutions to the most of the patterns. Other comparisons have also been done e.g. with RAD and UML (Odeh et al. 2006) where the conclusion was that translation from RAD to UML 2.0 Activity Diagram is likely to be feasible in particular cases, but it will rely on the ability of the translators to establish and maintain the equivalence between the two notations. In a study made by Kawalek (cited in Ould and Huckvale, 1995) he argue for that some modellers think it might be valuable to have a combination of notations at different levels when describing software processes.

3.1.1 Quality in Process Modelling

In today's literature there are few articles that discuss the question about quality in software process models (Moody et al. 2002.) The process models are according to Lindland et al. (1994) important in early development phases of information systems. A framework for evaluating the quality in modelled processes is proposed by Lindland et al. (1994). In this framework they identified three main factors that to be aware of when analysing a process model. The three main factors are; *Syntax quality* which represents how well the model and the common rules for the notation in the chosen modelling language correspond to each other. *Semantic quality* represents how well the model of processes corresponds to the 'real world' (the domain). *Pragmatic quality* that is about the use and understanding of the model, how usable it is and how well the users (anyone involved) will understand it. Lindland et al. (1994, p.45) is also mentioning three other factors that can influence quality on a model; *Language-domain appropriateness* which is about how well the language fit to the domain. Are there expressions needed to describe the domain? *Language-audience appropriateness* that is about to the extent the user of the model can understand, learn and use the language of the model. *Audience-domain appropriateness* that is about to what extent the audience already is familiar with the domain. All the factors can be seen in Figure 18.

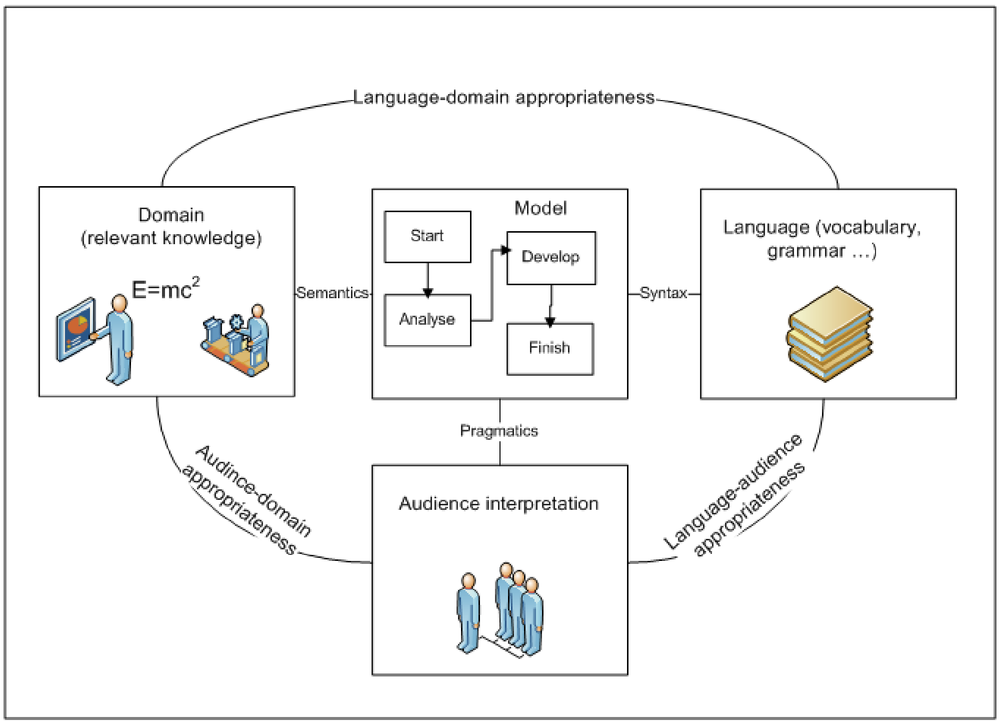

Language-domain appropriateness
Domain
(relevant knowledge)
E=mc²
Semantics
Model
Start
Develop
Analyse
Finish
Syntax
Language (vocabulary, grammar ...)
Pragmatics
Audince-domain appropriateness
Audience interpretation
Language-audience appropriateness

Figure 18. Quality factors according to Lindland et al. (1994, pp.44)

In order to reach a good quality Lindland et al. (1994) stated goals for each of the factors mentioned and they also suggested activities (means) to do for achieving the goals.

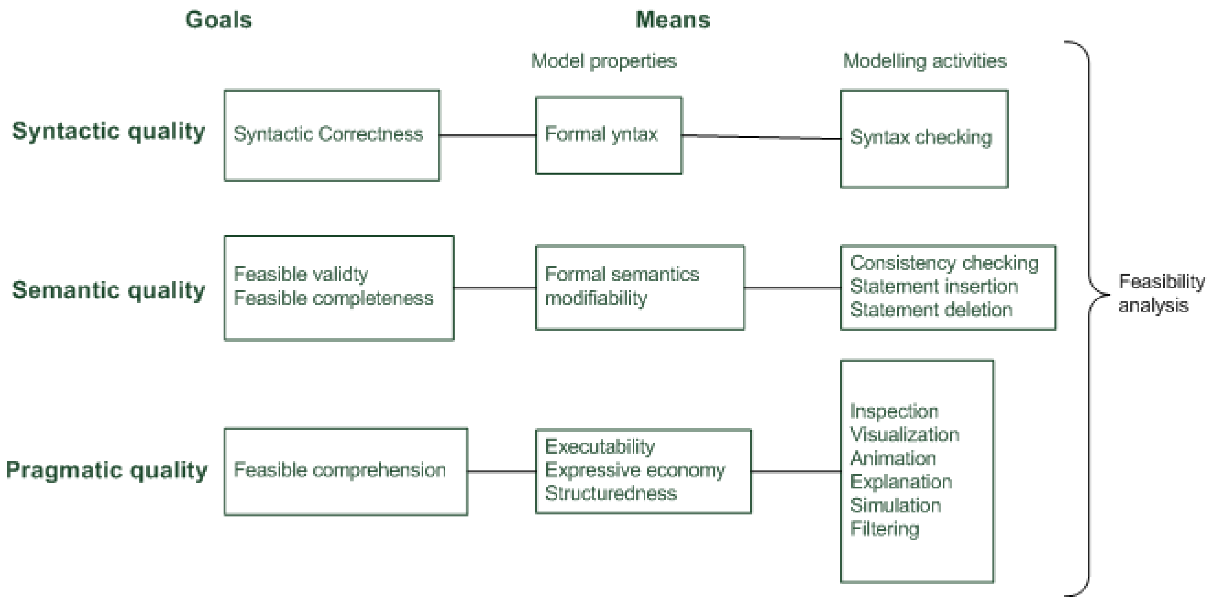


Figure 19. Quality factors with goals and means by Lindland et al. (1994, pp. 45)

According to Lindland et al. (1994) *syntactic correctness* is the goal for achieving syntactic quality. To achieve *semantic quality* the goal is to have a feasibly validity and feasible completeness. Validity means that all the statements made by the model are correct and relevant to the problem. Completeness means that the model contains all the statements about the domain that are correct and relevant. It is hard to achieve total completeness so Lindland et al. (1994, p.46) states that "the time to terminate a modelling activity is thus not when the model is perfect (which will never happen) but when it has reached a state where further modelling is less beneficial than applying the model in its current state". To achieve pragmatic quality the goal is to reach feasible comprehension and the means for this are executability, expressive economy and structuredness. Feasible comprehension is about that all parts involved should be able to understand the model. Since total understanding not can be achieved (e.g. large models) it is enough that feasible understanding is achieved. There is other means to achieve quality such as different modelling activities mentioned by Lindland et al. (1994) e.g. consistency checking and simulation, see Figure 19.

In order to evaluate process models Teeuw and van den Berg (1997) has developed a framework in which they propose nine criteria‘s divided in four dimensions (general (e.g. the price of a tool and customer support), BPR trajectory, functionality and "ease of use") that are usefully when evaluating process modelling concepts. The dimensions General and BPR trajectory are only important when purchasing a tool and not for evaluating therefore these two are not needed when evaluating models. The first criteria of the five in the dimension functionality is *expressive power/descriptive power* is about how easy it is to modelling aspects like reactivity (relationships between activities), data, data manipulation, repetition, time and probabilistic behaviour. Comparisons made with Lindland et al. (1994) shows that this criterion can be comparable to Lindland et al's (1994) criteria Semantic quality. The second criterion is *structuring* which imply at what degree the concept offer structuring techniques such as composition and decomposition, abstraction and refinement, and modularity and encapsulation? Comparing this criterion to Lindland et al. (1994) this can also be found in Semantic quality. Third criterion is *formal and methodological support* - does the concepts have a formal foundation (syntax and semantics), and is it accompanied by a method to construct models? Formal support can be found in can be found in the mean formal syntax according to Lindland et al. (1994). The methodological support doesn't have a corresponding area in to Lindland et al's (1994) framework. Fourth criterion is *possibilities for analysis* which intentions are what types of (functional or quantitative) analysis that theoretically can be performed on a model. There are similarities to semantic quality according to Lindland et al's (1994) framework in this criterion. The fifth and last of the criterions in this dimension is *relevance of concepts* that is about how appropriate are the offered concepts in the context of modelling (business) processes, and how generic are they? Compared to Lindland et al's (1994) framework this criterion has similarities with the goals feasible validity and completeness.

A modelling concept not used is worthless and spoiled work therefore Teeuw and van den Berg (1997) propose an "ease of use" dimension is important and the last criteria's are connected to the usability. *Accessibility*: are the concepts and modelling methods comprehensible, is documentation available and sufficient? This criterion can be found in feasible comprehension in Lindland et al. (1994). *Usability* describes how easily a process can be modelled, if the language environment does offer pre-defined constructs, libraries of high-level concepts etc and has similarities in Lindland's goal feasible comprehension. The third criteria *adaptability* is about how easily the concepts can be adapted to individual needs and compared to Lindland et al (1994) it has similarities with pragmatic quality. Final criteria *openness* describes if a language or tool based on these concepts can be used in combination with other languages and tools and also this criterion have similarities with to Lindland et al's (1994) pragmatic quality.

4 Method

In this chapter the methods used in order to conduct the present study are presented. The study embrace the review of the two main questions: (1) what role the described software process serves and (2) what quality attributes are important to the specific organization in the described software process. It also embraces a review of the use of tools for modelling software processes. The purpose with this chapter is to present the participants, the methods, the research instruments and the data gathering process in order to attain a high validity and reliability of the study.

An extensive literature review was conducted to find information and earlier research done on major software process modelling and standard notations. Parallel ongoing with the database searches a review of information available at the Internet was also conducted. This was primarily done with the search engines Goggle and Wikipedia. The databases , Institute of Electrical and Electronics Engineers database (IEEE Xplore), Science Direct database (including Elsevier), Association for Computing Machinery database (ACM) and Springer Link database were searched with combinations of the following keywords; modelling, software, diagram, workflow, visual, notation, diagram, business, description, activity, process, definition, language, software, adaption, and tailoring. The same key words were also used when searching the Internet with the Goggle search-engine and Wikipedia for information.

4.1 Participants

The work as presented in this study comprises employees at senior management level and employees at senior engineer level representing all disciplines in the development process e.g. software and hardware integrators at Ericsson Lindholmen (Sweden). All together 12 participants were invited to take part in the study. All 12 participants accepted to participate in the study. The participants were chosen by the author and the supervisors based on the will to map a good combination of managers, senior development engineers and software process engineers.

4.2 Design

In this study both quantitative and qualitative data was gathered. The quantitative data was the basis of the study in evaluating the believed roles of described software processes, high quality in described software processes and the need for tools and their importance in describing or modelling software processes. In order to provide a more depth to the study qualitative data gathered during the interviews was used. The interviews where performed face to face during a period of two weeks.

4.3 Research Instruments

The instruments used in the study were questionnaires together with face to face interviews. The questionnaire consisted of three parts and it can be seen in Appendix D. First part consisted of six statements regarding the interviewees view on the reasons of having a described software process in a software development organisation. All the statements were graded in a six-degree scale from strongly disagrees to strongly agree. The second part consisted of nine statements regarding the attributes for reaching high quality in a described software process. Also these statements were graded in a six-degree scale ranging from strongly disagrees to strongly agree. In the third and last part the interviewees was asked in an open question to name all tools that they knew about or had used in modelling software processes. The participants were also in this part asked to value the importance of such tool in modelling software processes with the same scale as used earlier. In each of the statements the interviewees where asked why they for instance had replied strongly agree, what had influenced their answer. They were also asked to prioritize the most important statements if several statements were given the same degree. As a final question the participants was presented four illustrations in different notations describing the same software process and they were asked to suggest the one that they intuitively preferred.

4.4 Procedure

First step in the study was to define and delimit the problem. After this had been done literature written about standard notations and software process modelling was reviewed. To get empirical data we used a questionnaire which was based upon information from the theoretical framework. The questionnaire was designed and approved at a meeting in February. The questionnaire was written in English but the interviews were held in Swedish. Each of the identified interviewees was selected by the author's supervisor Anna Börjesson. After the selection the interviewees the author contacted each person proposing date and time for an interview and all of the interviewees accepted. All interviews were done over a period of two weeks in February/Mars and lasted for an hour. At all the interviews field notes were taken and data was later manually analysed by the author. The reviewed literature and assessment of described software processes are used in the empirical study and this lead to those findings could be identified. By comparing findings identified in the empirical study with reviewed literature we could analyse and discuss the result and draw conclusions.

5 Result

This chapter presents the results given in questionnaires (quantitative) and interviews (qualitative). The first section begins with presentations of diagrams based on the answers given by the interviewees in questionnaires. All grades given in each statement in the questionnaires the scores were summed and diagrams based on the answers were created. The result chapter continues with presenting the qualitative result from the interviews that was conducted in the study.

5.1 Quantitative Data

The following two sections will present the result from the quantitative data gathered the questionnaires given during the interviewees. The first section presents the quantitative data given about the roles for the described process and the second section presents the quantitative data in the attributes for reaching high process quality.

5.1.1 Roles of the Described Process

This section describes how the 12 interviewees graded importance for reasons to use described software processes. The six-degree scale given for each statement was in range from strongly disagrees to strongly agree. All grades for each role given by the interviewees were added and a diagram created based on the answers that can be seen in Figure 20. Legend for staples in the diagrams is presented below.

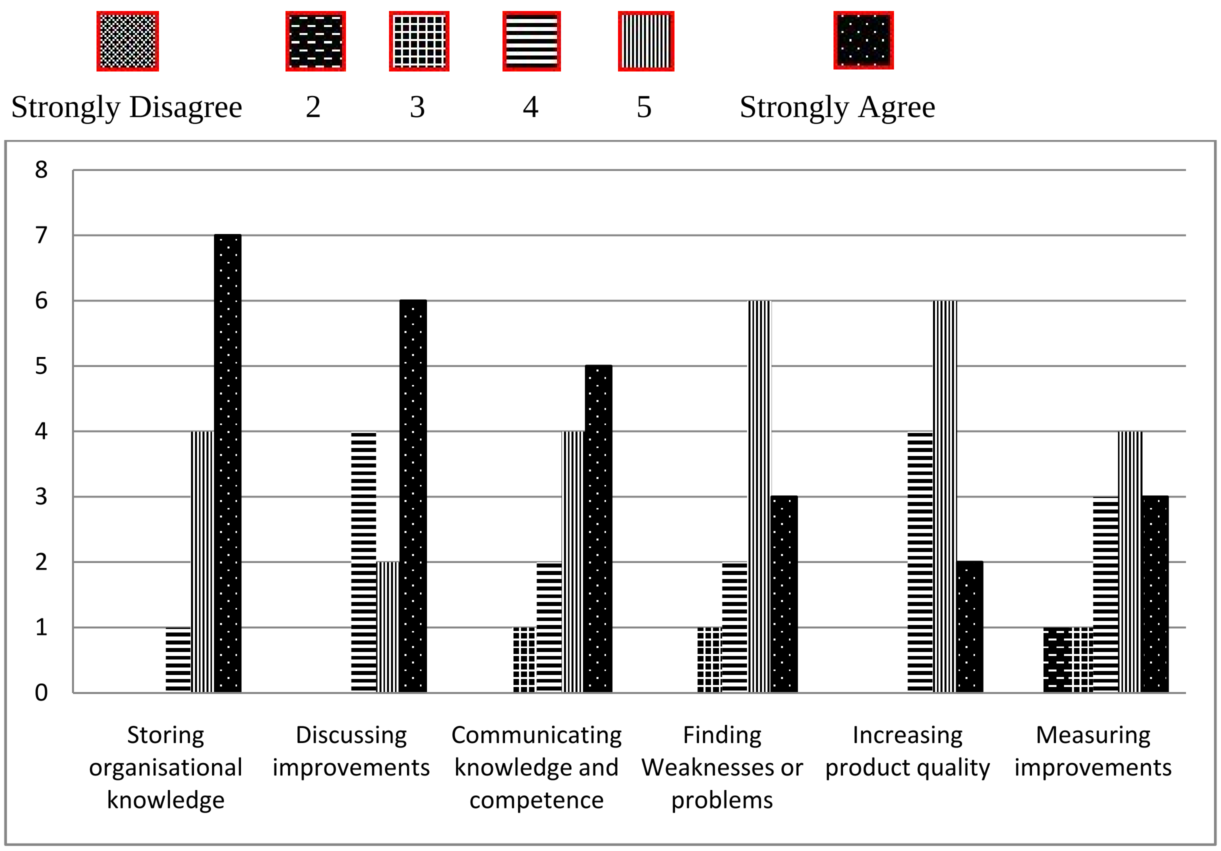


Figure 20. Roles of the described process

The result in Figure 20 shows on a strong support in ‘storing organisational knowledge’ as a reason for using described software processes due to that more than 50% of the interviewees strongly agreed on the reason and more than 30 % almost strongly agreed for the reason. As the second and third reason for using described software processes are ‘discussing improvements’ and ‘communicating knowledge and competence’. The least important reason for using described software processes is ‘measuring improvements’.

5.1.2 High Process Quality

This section describes how the interviewees valued attributes that make a high quality in described software processes. The scale given for each statement was from strongly disagrees to strongly agree. All grades for each attribute given by the interviewees were added and a diagram created based on the interviewees answers that can be seen in Figure 21.

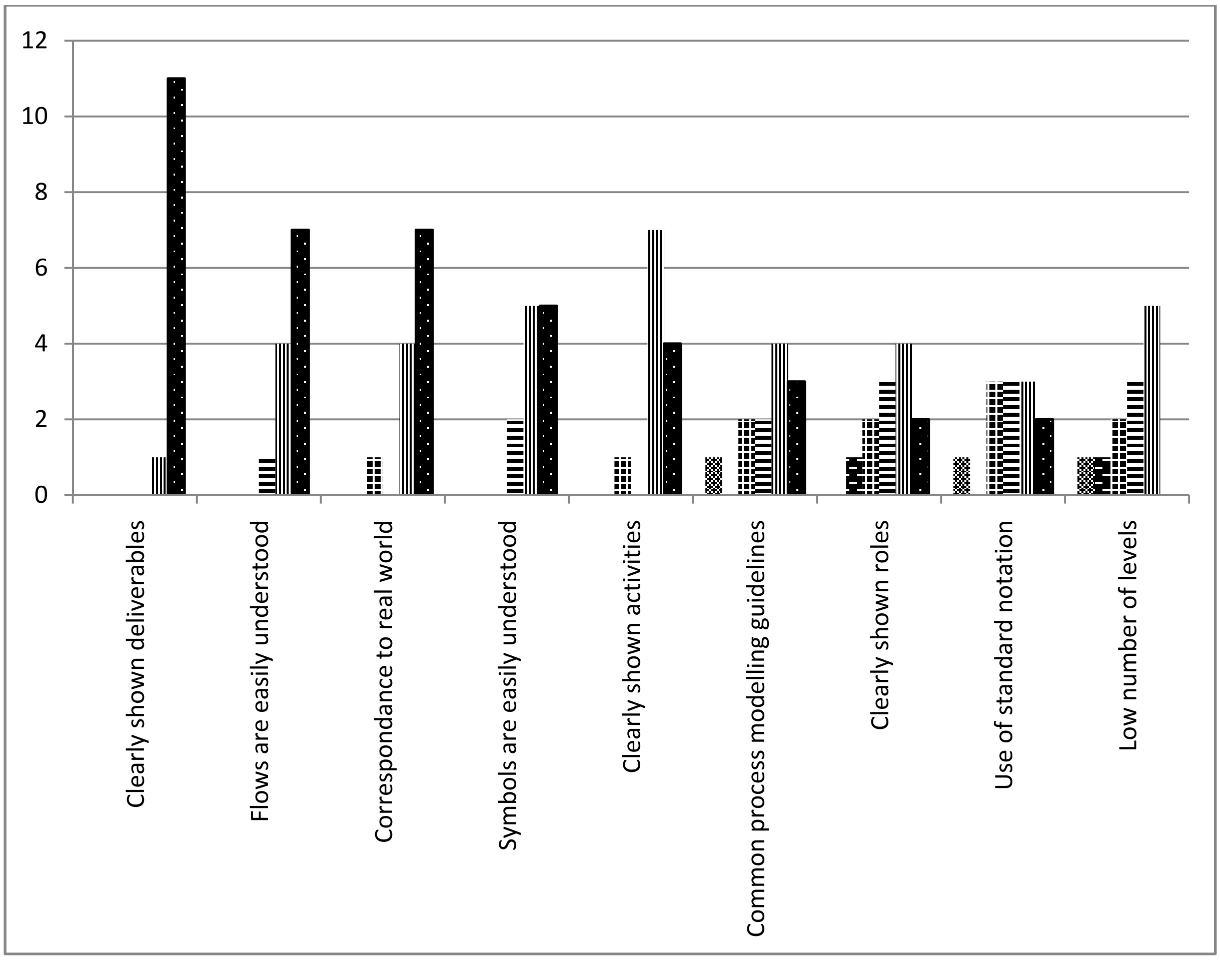


Figure 21. Importance of attributes for high quality in described processes

As the far most important attribute for reaching high quality in described software processes are clearly shown deliverables due to that more than 90 % strongly agreed on its importance. As the far most unimportant attribute for achieving high quality in described software processes are a low number of levels.

5.2 Qualitative Data

The following four sections will present the qualitative result from the interviews performed. First section present the result handling the reasons for having a described software process and the second section will present the result from the interviews regarding the attributes for reaching high quality in described software processes. The third section present the result from the interviews regarding tools used for modelling and the interviewees opinion of the importance in usage of tools. The fourth and final section presents the result regarding the preferred notation among the interviewees.

5.2.1 The Role of Described Software Processes

The interviewees see many roles for describing software processes and, in an open question the interviewees mentioned 11 new different reasons for using described software processes in addition to those we identified in the literature review. There is a strong agreement among the interviewees for primary reasons in describing software processes as a way for storing organisational knowledge, followed by discussing improvements, and communicating knowledge and competence. Several of the interviewees speak of these three reasons as a combination where it all starts with storing of the organisational knowledge. Respondent 8 stated "it starts with that the software process needs to be stored, somewhere in order to be able to be used for communicating knowledge. If used as the later it is possible to find weaknesses and problems and with this the discussion for improvements follows". When organisational knowledge is stored it is later used for discussing improvements and communicating knowledge and competence. For instance both Respondent 7 and 8 sees that storing organizational knowledge and communication of knowledge and competence are connected to each other in that they will lead to discussion of improvements.

While the interviewees largely agree on the usefulness of described software processes as a means for storing organisational knowledge, experience has taught the participants of the risks involved when time pressure otherwise may tempt improvements to be initiated solely based on weaknesses or problems found rather than first mapping these issues with the bigger organisational picture. Aside from this risk of sub-optimization, Respondent 5 also reflects on how large scale use of stored organisational knowledge may kill creativity among employees and thus argues for a balanced use of this stored knowledge. There are also problems in getting people to read other peoples documents in order to learn, this according to Respondent 4.

Regarding the role for discussing improvements, 7 out of 12 interviewees see various aspects of using described software processes as baselines for improvement. Primarily, a well described software process is argued to provide a common language and shared view for discussions. Respondent 1 argues that it is easy to take for granted that we talk about the same things when in fact we talk about different things. It is first when the software process is described we can be sure we talk about the same thing. Described software processes also play a role when initiating change (provides a given starting point) and enables comparison with actual events. Both Respondents 5 and 10 stress the importance to describe the software processes in order to have a base level so that it is possible to know what to improve against.

Finally, described software processes are by one interviewee also seen as hands-on specifications for what to do.

In the role for communicating knowledge and competence, if we ignore one interviewee who discusses software processes more on a project level than on organisational level and thus ranks this construct low, the elaborations given by the participants talk strongly about the value in using described software processes to communicate knowledge and competence. Respondent 2 captures the essence of this by arguing that the described software processes provide legitimacy for the communication. There is, however, knowledge that can’t be transferred by reading described p software rocesses and therefore person-to-person transfer is argued to add the last bit of knowledge transfer by Respondent 5. Respondent 4 provides a slight warning, though, that it is often easy to talk about experiences, but hard to actually see people listening and making use of this. “Today we are not so good in storing organizational knowledge – it is boring to read documents created by other people, the problem is that it is easier said than done to adapt the information” states Respondent 4.

When it comes to the role of finding weaknesses and problems, using root cause analysis to improve the described software processes is of major importance, especially for those working with Japanese customers. Two interviewees talk, however, about already knowing problems prior to describing software processes. This appears to be a sign of maybe not having all software processes quite as well established as in the case of the Japanese customer. One of these interviewees (Respondent 6) argues that the next step for handling known problems is not necessarily describing the software process. Instead, he argues for that the weaknesses are found pretty easily since they are clearly visible; there is no need to use a described software process to find these. This argument is contrary to the risk for sub optimization that Respondent 8 argues strongly about. This respondent (8) argues for that it is only when using a described software process description in order to find a weakness or a problem these can be found, otherwise the risk is high for sub optimisation.

The participants do not see a described software process to be used for measuring improvements, instead they see other things. Two of the participants stress the need for the data to be used when measuring to be repeatable.

Further, the participants don’t see the role for describing a software process as major driver for increasing product quality. There are according to several interviewees the skills and knowledge that employees possess that decides product quality, not the described software process. Respondent 10 argues that the product quality is decided by employees that in the end make the difference. According to the respondent, this is due to that a limit in how much and how detailed instructions that a person can capture until it suffocates the creativity and in this way may decrease the product quality. According to Respondent 9, too little efforts are spent in described software processes on the “soft values”. Soft values are according to the respondent defined as the attributes that makes people work effective together in a project group. When roles are appointed in projects it is often up to the ‘discipline managers’ to see to that the “soft values” are taken care of. One of the interviewees (Respondent 11) claimed that to reach an increased product quality, it is better to focus on describing the product instead of processes, in this way it is easier to make improvements. This respondent said that his department is now trying to use an Agile way of working where they put more focus on the product than on processes.

5.2.2 Quality in Described Processes

In order to reach high process quality, the interviewees see the flow where deliverables, activities and roles should be clearly shown. The interviewees strongly agreed that the most important for having a high potential of quality in described software processes is a clear view of deliverables. This is far highest graded attribute of all that the interviewees valued. All interviewees apart from one (this interviewee had it graded as five in the six-degree scale) strongly agreed to the importance of clear and visible deliverables. Respondent 10 claimed "when deliverables are made visible in a flow it is possible to find where it is missing deliverables which has been proven. With described software processes it becomes obvious what shall be achieved with what deliverables to people it is also possible to use deliverables as a measurement in progress". Another interviewee (Respondent 2) agrees and claims "that the deliverables are clearly shown is very important for software process quality; it is in the interfaces between deliverables that product quality is created and the deliverables should be well defined in order to reach high quality in software processes".

Some of the interviewees also talked about visible and clear roles as an attribute for high quality in described software processes but only two of the interviewees strongly agreed to this attribute and the spread in grades is large while the overall grade is second lowest of all attributes the interviewees had to value importance of. Respondent 12 argued a problem with a role-focus as "it might be a benefit that it is possible to say who shall do what but it might also be a drawback since the flows in a certain discipline can be connected more to an individual (or group) rather than a role". This will lead to that how the organization is set-up will rule on behalf of requirements. Respondent 12 continued arguing that "focus on roles often give negative effect in that often when we start projects we begin with plan what roles we need and not what shall do. We don't have the free mind set about what roles are in software processes instead we locks a role to a certain position and this makes more damage than benefit". This view on risks with roles is also supported by Respondent 8. According to Respondent 7 roles come naturally when identifying activities and deliverables and this opinion is also supported by Respondent 1. An example of this is verification where the activity "verification" first is identified and this leads to that a role "verifier" is created. In the hardware discipline synchronization roles have been created that follow the deliverables in time. An example on this is a role "board-responsible" that follow a circuit-board over time from creation to maintenance and synchronize different disciplines that are connected to the creation of the board. Respondent 7 concluded that "it is nice that the roles are visible but the most important is activities and that their input and outputs are visible".

A low number of levels are the attribute which is lowest graded of all attributes by the interviewees in order to reach a high quality in a described software process. The common opinion by the interviewees seems to be that there need to some kind of trade-off in how many levels that should be included in a described software process. Respondent 9 states that when it comes to the question about number of levels/clicks in order to reach information there must be a trade-off, it has to do with what the organisation will manage. If there is not relevant and enough information on the higher levels the process will not be used. There must be enough keywords on the higher levels to attract the users to use the software process. One of the interviewees argues that it’s a stakeholders’ concern and if there are too few levels, there is a risk that important information is missed. There is a need to take in consideration, who that shall use the described software process and what degree of experience this user has. In a model or description that has too many levels there is a risk to get stuck in a detail level and it is hard to get the ‘big picture’. The risk if the levels are flattened out is that the flow becomes lost, leaving the users with a feeling of ‘too many arrows’ to make the flow visible. According to respondent 10 this quality attribute might be the hardest to achieve.

The importance in that flows are easily understood is graded second most important closely followed by the correspondence to the real world. According to one interviewee (Respondent 7), in a project the hardware discipline follows one flow and the software discipline follows another flow, and these disciplines do not use the same vocabulary. People ask bout information from the software process in how to use this usually because we miss to show overlaps and increment in the flows therefore the reality often doesn’t match the described software process. Both Respondents 5 and 11 claims a importance of the flows in that if a user can’t understand the flows in a described software process this will function as a threshold and he or she will not then be able to understand anything at all of the described software process. Respondent 4 states that the flow is important since “it tells us what comes out in the end, what the result is”.

The described software process correspondence to the real world influence as a quality attribute are according to the participants up to who is using the described software process. The importance is higher graded when it comes to new employees or inexperienced people.

When it comes to a standard notation and common modelling guidelines the interviewees grade the importance of these low. Some of the interviewees see these as a form of restriction that might suffocate people’s creativity if they are too rigorous and on the other side some of the interviewees see it as mandatory that there is a restriction in how to describe or model a software process. Since some of the interviewees argue that product quality comes from the creativity people posses in solving problems, if users get too restricted from common modelling guidelines in their work with software process modelling the product quality will decrease since creativity will be suffocated. Respondent 8 on the other hand sees a common software process modelling guideline as a support in to creating flows that are easy to understand. A standard notation is seen by several interviewees as a way to have a common vocabulary in the organisation. According to Respondent 9 “it is only when standard notations is used that users can talk the same language and understand each other. Usually there is a problem when different disciplines talk to each other (i.e. hardware and software disciplines).” Two of the respondents (2 and 7) see a standard notation as the next step after understanding of symbols and flows is reached. There are many other things to work with before you start work with the notation. One of the interviewees strongly disagrees to a standard notation with the argument that often we follow RUP (the interviewee sees RUP as a standard notation) but sometimes we do deviations from this and it works anyway therefore it can’t be so important. The interviewee however sees that it has obvious benefits using a standard notation i.e. to transfer knowledge to new employees.

5.2.3 Tools and the Importance of them

On an open question to name tools that the interviewees have come in contact with, used or have heard that are used for describing or modelling software processes, the interviewees named 14 variants of tools. This gave the result with the 14 variants and the six most named tools were Rational Method Composer (11), Microsoft PowerPoint (6), HWDP (5) (an Ericsson developed tool), HTML (4), PLCM (3) (the method how Ericsson centrally describe the software development process), and Microsoft Visio (3). The most mentioned tool was RMC developed and sold by IBM. The tool RMC was mentioned by all apart from one of the interviewees.

The interviewees agree strongly, that it is of high importance to use a tool in describing and modelling software processes. This is showed in that more than half of the interviewees strongly agree on its importance of using a tool for describing or modelling software processes. This depends on that the interviewee's sees today's described software processes as very complex and withholds a large number of deliverables and roles. This leads to that the information that can be extracted from the processes is detailed and intense. To support this Respondent 7 stressed "today we have so advanced and complex software processes that we must use a high-quality tool to be able to control them, Microsoft PowerPoint won't do the job anymore and we cannot do it on free-hand." In order to check that all deliverables and roles are essential and to find potential gaps or discrepancies in the need of deliverables or roles, a tool making this visible is very much needed. That a tool needs a consistency check function is of most importance if it is to be used for describing complex software processes, this opinion are stressed among several of the interviewees. Respondent 1 argues that tools give us good opportunities to find gaps in missing or unnecessary roles, or troubles with deliverables. With the use of a tool that brings all to a total unit (a whole) it will be possible to find these gaps. This is self experienced from the latest project the respondent was involved in. In this project they used RMC when describing a software process where it became clear to the project that deliverables was missing. Many interviewees spoke about using Microsoft PowerPoint or Microsoft Visio for describing software processes but only in presentations because these tools cannot be used describing software processes that are to be used in daily work because they lack the consistency check on deliverables and roles, and therefore should not be used in other ways than for presentations. This is summed by Respondent 10 who said that "when using Microsoft PowerPoint though it is up to good reviews by the projects in order to find gaps in the deliverables or roles. With a tool such as Microsoft PowerPoint it is hard to keep updated described software process models and updated links to the processes".

Another aspect given is that tools for describing software processes are necessary but there is mandatory that we have the possibility to adapt the tool in to how we work daily and not the other way around. The tool is there to serve the organisation and examples on the opposite was discussed by Respondent 3 that also talked from earlier experience where the tools had steered how the daily work was done because an extra moment was added when the production teams tool couldn't use the material the design team created based on their tool.

5.2.4 Qualitative Review of Preferred Process Notation

When presented the same software process that describes the development of Alfa, an Ericsson product that is described in four different notations the interviewees preferred different notations describing different levels of the software process. The favourite notation to the interviewees when it came to the top levels was SPEM (UML Activity Diagram) and at the lower levels it was BPMN or HWDP (at document level).

The RAD was not a favourite among the interviewees; it was only one of the interviewees that argued for the favouritism for this notation. Several of the interviewees thought this notation had similarities with the SPEM notation and the later is preferred because of the inheritance in working with similar notation. Arguments against the RAD as notation among the interviewees are given by Respondent 11 “it is not intuitive to med how to interpret the symbols” and by Respondent 5 that states “this model is unclear, there are no obvious flows and it is not obvious where the end is.” Only one of the interviewee’s favourites this notation (after discussing for a long time) and this is at the third level. The interviewee sums when discussing the third level that “RAD becomes much better at third level because it has an intuitive role and flow description that can’t be found in the other notations (it has a high potential)”.

All participants except one instantly recognized the SPEM notation this due to that it looks the same as the well known UML notation with it is based on. A dilemma with Activity Diagrams like the one in SPEM is according to the interviewees to get a clear understanding when in time, and if activities shall be finished before the following activity can start, this is hard to communicate to the users with this notation. It is also difficult to show the iterative activities in the models. Today Microsoft PowerPoint is used to show and clarify the flows; this is a weakness in all models. Respondent 11 argue for that the models are too strict and forcing, more adaptability is needed – much should be more optional. According to this interviewee “we need more Agile-thinking in the process descriptions. Today we see more to the need of tools for describing software processes than to the business case”. Even at the third level the order of when activities should be executed and the deliverables used was unclear to the interviewees i.e. by Respondent 1 that stated “the order on the deliverables and the activities working with these is missing at third level”.

The BPMN notation was by a couple of the interviewees seen as to technical, it reminded of blueprints describing a circuit-board to them. The ‘plus signs’ in the activity boxes at the higher levels in the model was also irritating which caused this notation to become disliked by several interviewees. At the higher levels this notation is not a favourite, instead it is at the lower levels that it is one of two favourite and this is due to the clearly shown roles and clearly shown order in deliverables. Respondent 1 states that “the order of deliverables missing in SPEM model is well described in third level in this model”. Respondent 5 argue that “this model is good in that the flow among deliverables is clear. I have never seen it before but it is easy to understand the flows”.

Most of the interviewees recognized the HWDP notation presented to them quite fast. This notation is preferred as one of two at the lower levels due to the clearness of deliverables and the status of these. This model lacks the ability to see which role shall handle specific deliverable but this is not a problem to the interviewees. Respondent 1 sums the thoughts saying that the described software process is a placeholder for guidelines, templates and checklists.

6 Discussion

In this chapter we present the result from the study on described software processes and notations. First the result from analysing the knowledge of described software processes and tools is presented. This is followed by the result from analysing the roles of described software processes and the result of what is of importance for high quality in described software processes. The discussion chapter ends with suggestion for future research.

6.1 Knowledge of Described Process

From this study we have learned that there are no obvious standard notations or tools for software process descriptions. The literature argues there are four main modelling techniques, i.e. Activity-oriented techniques, Object-oriented techniques, Role-oriented techniques, and Speech-act oriented techniques. We decided, as described in chapter two, not to discuss Speech-act oriented techniques in this study as this approach is only vaguely described in the literature with rather few references. It is clear that none of (1) Activity-oriented techniques - focusing on the definition of business processes as a sequence of activities discussed by Owen and Raj (2005), and White (2004), (2) Object-oriented techniques - leveraging the more comprehensive modelling constructs of object-orientation to capture business processes discussed by Kalnins and Vitolins (2006), Haumer (2006), and Russel et al. (1994) or (3) Role-oriented techniques - modelling business processes based on the specific organisational roles and responsibilities involved discussed by Ould & Huckvale (1995), and Ould (1995) is dominated in the reviewed literature.

The interviews supported what was already known to us from the literature study. The interviewees had no common picture of a standard tool to use for modelling software processes (Davies et al. (2006). This was verified as all 12 of the interviewees gave 14 different variants of a software modelling tool, e.g. RMC, Microsoft PowerPoint, HMTL, and Microsoft Visio.

Our interviewees show, however, a considerable interest to both notations and tools for the described software process since they agree that there is of high importance to them to use a tool in modelling software processes. This goes against the findings of Iivari (1995) that found that perception of effectiveness of CASE-tools rather low. The favour for CASE-tools in this study is motivated with that today software processes are highly complex containing and usage of a high number of deliverables. In order to control all deliverables there is a need of a tool with consistency check. Some of the interviewees preferred simpler tools such as PowerPoint and Visio but due to the lack of consistency checking, they at the end argued for more advanced tools containing a consistency check, such as Rational Method Composer. Arguments given by interviewees for the importance in reaching high quality using described software processes with a low number of levels (few clicks) also support the need for more advanced tools. It is clear that the advanced tools are needed in order to handle a high number of levels in a correct way since this not will made with simpler tools like Microsoft PowerPoint.

After analysing the illustrations showing the software process for developing the product ALFA for a while almost all of the participants preferred different notations at higher levels compared to the lower levels. This is due to the clear presentation of deliverables at the lower levels in BPMN and HWDP. The interviewees want to see the deliverables, who they will get them from and to whom they shall deliver to. Respondent 11 argues for that BPMN "is the preferable notation because of that this is best on the third level and it clearly shows the flow of deliverables and who's using them". Also Respondent 6 support this when saying "this model is good in that the flow among deliverables is clear. I have never seen it before but it is easy to understand the flows". The interviewee's intuitive preferred notation shows that there is not one notation that suits all levels of a described software process. At higher levels the interviewees preferred SPEM/UML but at lower level they preferred Business Process Modelling Notation (BPMN). This supports Kawalek (cited in Ould and Huckvale, 1995) who in a study also found that there usually is a need for different kinds of notations describing different levels in a software process model. In order for the employees to make out what to do with what the deliverables need to be clear and visible. Because that different notation is good at different levels the preferred illustration became a combination of different notations by most interviewees. What to take notice about is that no one of the interviewees tended to have a high preference for a described software process in Role Activity Diagramming (RAD) which is in line with interviewee's non-focus on roles.

We summarize the general learning of knowledge of described software processes as "described software processes are of high interest, but there are no obvious standards for how to do it".

6.2 Roles of Described Software Processes

The role of described software processes is thoroughly discussed in research by Ould and Huckvale (1995), Ould (1995), Ellmer and Merkel (1996), and Curtis et al. (1992). While Ould and Huckvale (1995) argues for the use of described software processes as a focus for discussion, means for discussions and basis for analysis , Curtis et al. (1992) focus on the automation of described software processes. Common for Curtis et al. (1992) and Ellmer and Merkel (1996) are that they all argue for storing organisational knowledge in a repository forming an organisational memory.

Our interviews revealed a great interest in reasons for having described software processes in that we got 11 different roles on our open question, *I believe described processes are important for other reasons (which?).* The most highly valued reason to use a described software process was for 'storing organisational knowledge' closely followed by 'discussing improvements' and 'communicating knowledge and competence'. For instance one interviewee stressed that 'storing organisational knowledge' is a fundamental role for describing software processes, "[w]hen this is done it is possible to communicate knowledge and competence and these two fundamental roles will lead to that improvements are discussed". The opinion of 'storing organisational knowledge' as a fundamental role is supported by one interviewee who also connects the use of a 'stored organisational knowledge' to gaining market shares in a competitive market. According to the same interviewee, if the knowledge is stored it is later possible to show the customers what is done in error prevention and error correction. In this way customers trust is gained and it is possible to use it as an advantage to competitors and a way to win market-shares. Importance for storing of organizational knowledge in form of an organizational memory discussed by the interviewees is argued by Ellmer and Merkel (1996) and is supported in our study.

We summarize the general learning for roles of described software processes as "there are many good reasons for having described software processes and we conclude our findings to be in coherence with the reviewed literature".

6.3 Quality of Described Processes

High quality of described software processes is necessary for promoting actual use of the described process. The attributes used to describe 'qualities of described software processes' are quite many. Ould and Huckvale (1995) brings up that the described process must be intuitive and easy to read, have a real world correlation, relate to roles and handle complexity. Lindland et al. (1994) argues that syntactic (symbols and flows), semantics (real world correlation and pragmatics (actual understanding of users). Tuuew and van der Berg (1997) focuses on function and ease of use which is similar to Lindland et al's (1994) syntactic and semantics.

From our interviewees, we did however find a major interest of a quality attribute that renders little attention in the reviewed literature. Almost all interviewees strongly argued for the need of clearly shown deliverables in the described software process. Respondent 1 states "within 'Ericsson product' process the flow of deliverables are prioritized and usually the most common question. It is a strong focus among employees that the deliverables is visible in time tables". Respondent 9 claims that "the deliverables are of high importance - it should be clear what deliverables shall be transferred between activities". Further Respondent 9 argues that "it should in the flow be possible to see what to do, with what deliverables and in what time and this should not be placed in different documents". It is the deliverables that is used in work every day and it is of highest importance that an employee knows who is doing what with what; this is what Curtis et al. (1992) argues for. The deliverables are not a main focus discussed among either in Ould and Huckvale (1994), Lindland et al. (1994), and Tuuew and van der Berg (1997) where primary the first authors instead focus on importance of roles. The interview study revealed on the other hand a rather big uninterested focus for roles in described software processes which have a strong focus at Ould (1995). I.e. Respondent 8 argue for the risk in matching roles with a need of competence making it as a form of work description and Respondent 12 states that "focus on roles often give negative effect in that often when we start projects we begin with plan what roles we need and not what shall do. We don't have the free mind set about what roles are in described software processes instead we locks a role to a certain position and this makes more damage than benefit". That the roles are unfavoured is also shown when the participant's decided a preferred notation in that described software processes described in Role Activity Diagramming solely based on roles quite fast was rejected by the interviewees during discussions.

Closely related to the focus on deliverables, the interview study also reveals a major interest for interfaces between software processes (i.e. input and output) which also shows no attention in the reviewed literature. Several of the interviewees argue for that the product quality is dependent by the skills that the employees possess and not by described software processes. The product quality is then created in the interfaces between hand-over of deliverables between the employees and is influenced by the skills that the employees possesses. One of the interviewees stated that there today are roles created explicitly for supporting the hand-over's of deliverables during a products phases (different disciplines) in a development cycle. This interface-attribute isn't mentioned in any of the related literature reviewed.

Our study also shows that the importance for a low number of levels in a described software process is rated as the least important among the interviewees in creating high quality in described software processes. The interviewees show a strong disagreement about the importance of this, see section 5 and Figure 21. Arguments for a low number of levels are the same as for the correspondence to real-world in that a complex solution has many details which need a high number of levels to cover all details and not miss important information. The supporters against a low number of levels in described software processes sees a risk in having many levels in that users bookmarks lower levels and always use the bookmarks and misses changes that happens at higher levels, this is supported by Curtis et al. (1992) as a problem causing discrepancies between actual behaviour and a stated software process in an organisation due to that users don't look at high-level software processes. One interviewee thought the number of levels was irrelevant, it was more important with a stable structure so that it is possible to find the information needed at the same location every time.

We summarize the general learning of attributes for reaching high quality in described software processes as "there are differences between user's demands in what described software processes should present and what can be found in the literature".

6.4 Implication for Future Research

It is likely that the increasing interest in distributed and global software development will further increase the need of software process control. Many different individuals, distributed over many different countries, with different cultures and in different time zones working together to produce one release of a software product need to be coordinated. A described software process is one possible tool to manage this coordination. The described software process needs however to be interpreted in one common way to be useful. It is likely a world-wide software process notation standard could facilitate this need. UML has facilitated the understanding of software product descriptions. A similar standard focused on software process descriptions is likely to be useful for the body of knowledge in the area. We encourage students, researchers and practitioners to further study the role of software process descriptions and quality attributes for software process description. These types of studies will likely push for a standard for software process notations and descriptions.

Organisations working in a global software development environment with agile approaches will lead to a new focus in what is seen as reasons for using described software processes and attributes for reaching high quality in these describes software processes. A suggestion for future research is to analyse these reasons and attributes from an Agile point of view.

In our open question given to the interviewees to propose other reasons for using software process descriptions 11 new reasons not mentioned in the reviewed literature was given us.

An implication for future research is to analyse the importance to these new reasons contrary to the ones indentified in the reviewed literature.

7 Conclusions

One way to enable effectiveness in software development organisations is to describe software processes in coherent ways to enable understanding between different target groups using the same software process. The increasing interest in distributed and global software development has made the matter of described software processes even more important as well described (and used) processes increase control.

This study sets out by performing a literature review of current software process notations followed by an interview study with senior managers and senior engineers at an organisation working extensively on a global level. In doing so, we focused on two questions: what role the described software process serves, and what quality attributes are important to the specific organisation in the described software process?

The contributions made from this work are:

(1) Overall support for the many different reasons for describing software processes. However, we can see a clear focus on priority of three roles; like ‘storing organisational knowledge’, ‘discussing improvements’, and ‘communicating knowledge and competence’.
We also notice a risk in using to strict modelling guidelines or standard notations in that it might lead to a decreased product quality since it will suffocate employee’s creativity in finding the best solutions.
In addition, it is noteworthy that the participants in the study added 11 (!) additional reasons for using described software processes.

(2) In terms of attributes for achieving high quality in describing software processes, the study reveals two surprising findings, as roles are argued well against by several interviewees (i.e. contrary to what our literature review suggested) and that clearly and understandable deliverables are strongly argued for by all interviewees (not mentioned in the reviewed literature). Also, it appears that low number of levels is not an obviously desired criterion in reaching high quality in a described software process.

(3) The literature review in itself can be considered a third contribution made particular as this study constitutes a richer and more complete mapping of different process notations, and the roles and quality attributes that are relevant to the area.

(4) Finally, this work also contributes by elaborating on what will be of importance for future research in order to respond to the once again increasing need to understand the interplay between control and agile oriented approaches for software development.

8 References

Abrahamson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. (2003). ’New Directions on Agile Methods: A Comparative Analysis’ *IEEE Computer Society*, pp. 244-254.

Acuna, T., S., and Ferré X., (2001). 'Software Process Modelling', *In Proceedings of the World Multiconference on Systemics, Cybernetics and informatics: information Systems Development-Volume I - Volume I*, pp. 237-242

Ambler, S. W. (2005). *The Elements of UML 2.0 Style*. New York, USA: Cambridge University Press.

Arlow, J., Emmerich, W., and Quinn, J. (2004). Literate Modeling - Capturing Business Knowledge with the UML. *Lecture Notes in Computer Science* , 1618 (1), pp. 189-199.

BPMN, (2007), OMG Specifikations [Online]. Available at: http://www.omg.org/technology/documents/spec_catalog.htm (Accessed: 12 January).

BPMI, (2007). Business Process Management [Online]. Available at: http://www.bpmi.org/ (Accessed: 12 January 2007).

Connel, C. J. (2001). *SW-CMM*. [Online]. Available at: http://www.chc-3.com/cs511/fall2001/lectures/cmm_intro.htm Accessed: 1 February 2007).

Curtis, B., Kellner M., I., and Over, J., (1992), 'Process Modeling', *Communications of the ACM*, 35(9), pp. 75-90

Davies, I., Green, P., Rosemann, M., Indulska, M., and Gallo, S. (2006). ‘How do practitioners use conceptual modelling in practice?’, *Data & Knowledge Engineering*, 58 (1), pp. 358-380.

Ellmer, E., and Merkl, D. (1996). Considerations for an Organizational Memory in Software Development. Software Process Workshop, 1996. *‘Software Process Workshop, 1996. 'Process Support of Software Product Lines'., Proceedings of the 10th International’* , 1(1), pp.60-62.

Floyd, C., (1986). ‘A Comparative Evaluation of System Development Methods’. *Information and Software Technology*, 37(2), pp. 119-126.

Haumer, P. (2006). *Software & Systems Process Engineering Meta-Model (SPEM 2.0)*. Available at: http://www.omg.org/cgi-bin/doc?ad/06-11-03 (Accessed: 10 January 2007).

Holt, J. (2004). *UML for Systems Engineering: Watching the Wheels*. E-book [Online]. Available at: http://library.books24x7.com.proxy.lib.chalmers.se (Accessed: 7 January 2007).

Jäger, D., Schleicher, A., and Westfechtel, B. (1999). 'Using UML for Software Process Modelling'. ACM SIGSOFT Software Engineering Notes', *ACM SIGSOFT Software Engineering Notes* , 24 (6), pp. 91-108.

Iivari, J., (1995). 'Factors affecting perceptions of CASE effectiveness', *IEEE Software* 4, pp. 143-158.

Kalnins, A., and Vitolins, V. (2006). 'Use of UML and Model Transformations for Workflow Process Definitions', [Online] Available at: http://arxiv.org/abs/cs.SE/0607044 (Accessed at: 16 January 2007)

Katzenstein, G., and Lerch, J. F. (2000). 'Beneath the Surface of Organizational Processes: A Social Representation Framework for Business Process Redesign'. *ACM Transactions on Information Systems* , 18(1) , pp.383-422.

Kueng, P., Bichler, P., Kawalek, P., and Schrefl, M. 1996. How to compose an object-oriented business process model? In *Proceedings of the IFIP Tc8, Wg8.1/8.2 Working Conference on Method Engineering on Method Engineering: Principles of Method Construction and Tool Support: Principles of Method Construction and Tool Support* (Atlanta, Georgia, United States). S. Brinkkemper, K. Lyytinen, and R. J. Welke, Eds. Chapman & Hall Ltd., London, UK, 94-110.

Lindland, O., Guttorm, S., and Sölvberg, A. (1994). 'Understanding Quality in Conceptual Modelling', *IEEE Software,* 11(2), pp. 42-49.

Lyalin, D., and Williams, W. (2006). 'Practical Considerations when Selecting a Notation for Business Modeling', [Online], Available at: http://www.bptrends.com/publicationfiles/09-06-ART-Counterpoint-LyalinWilliams1.pdf . (Accessed: 16 January 2007).

Mcmanus, J., and Wood-Harper, T. (2003). *Information Systems Project Management: Methods, Tools and Techniques.* Harlow: Prentice Hall.

Moody, D. L., Sindre, G., Brasethvik, T., and Sölvberg, A. (2003). 'Evaluating the Quality of Information Models: Empirical Testing of a Conceptual Model Quality Framework', *IEEE Computer Society*, pp. 295-305.

Murdoch, J., and McDermid, J. A. (2000). 'Modelling Engineering Design Processes with Role Activity Diagrams'. *Journal of Integrated Design and Process Science*, 4 (2), pp. 45-65.

Nandish, P. V. (2000). 'Healthcare Modelling through Role Activity Diagrams for Process Based Information Systems Development'. *Requirements Engineering*, 5 (1), pp. 83-92.

Necco, C., R., Gordon, C., L., and Tsai N., W., (1987) 'Systems Analysis and Design: Current Practises', *MIS Quarterly (Dec.)*, pp. 461-475

Nerur, S., Mahapatra, R., and Mangalaraj, G. (2005). 'Challenges of Migrating to Agile Methodologies'. *Communications of the ACM* , 48 (5), pp. 73-78.

Odeh, M., Beeson, I., Green, S., and Sa, J. (2006). 'Modelling Processes Using RAD and UML Activity Diagrams', [Online] Available at: http://www.cems.uwe.ac.uk/~sjgreen/RAD&AD_V2.pdf, (Accessed: 16 January 2007).

OMGa, (2007). Object Management Group [Online] Available at: http://www.omg.org/ (Accessed at: 12 January 2007)

OMGb, (2007). Object Management Group [Online] Available at. http://www.uml.org/#Links-UML2Tools (Accessed at: 12 January 2007)

Ould, M. A. (1995). *Business Processes*. Bath, United Kingdom: John Wiley & Sons Ltd.

Ould, M., and Huckvale, T. (1995). *Process Modelling - Who, What and How; Role Activity Diagramming*, in W. Kettinger, & V. Grover, *Business Process Change: concepts, methods and technologies.* London, United Kingdom: Idea Group Publishing, pp. 330-349.

Owen, M., and Raj, J. (2005, July 26). 'BPMN and Business Process Management'. [Online] Available at: http:// www.telelogic.com (Accessed: 16 January 2007).

Paulk, M. C., Curtis, B., Chrissis, M.-B., and Weber, C. V. (1993). 'Capability Maturity Model, Version 1.1.' *IEEE Computer Society Press*, 10 (4), pp.18-27.

Russel, N., ter Hofstede, A. H., van der Aslst, W. M., and Wohed, P. (2006).'On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling', *Australian Computer Society*, pp. 95-104.

Siegel, D. J. (2007). Introduction to OMG´s Unified Modelling Language. . [Online] Available at: http:// www.omg.org/gettingstarted/what_is_uml.htm (Accessed: 16 January 2007)

Teeuw, W. B., and van den Berg, H. (1997). On the Quality of Conceptual Models. [Online] Available at: http://osm7.cs.byu.edu/ER97/workshop4/tvdb.html (Accessed: 20 January 2007)

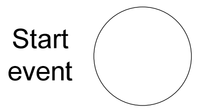
Tilley, S., and Huang, S. (2003). 'A Qualitative Assessment of the Efficacy of UML Diagrams as a Form of Graphical Documentation in Aiding Program Understanding', *ACM Press*, pp. 184-191.

Wang, W., Ding, H., Dong, J., and Ren, C. (2006). 'A Comparison of Business Process Modelling Methods'. *IEEE Xplore* , pp. 1136 - 1141.

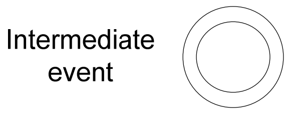
White, S. (2004). 'Process and Modelling Notations and Workflow Patterns'. [Online] Available at: BP Trends: http://www.bptrends.com/publicationfiles/03-04%20WP%20Notations%20and%20Workflow%20Patterns%20-%20White.pdf (Accessed: 15 January 2007)

Wikipedia, (2007). webServices [Online]. Available at: http://en.wikipedia.org /wiki/Web_services (Accessed: 12 February 2007).

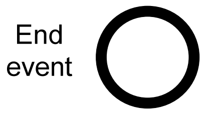
9 Appendix A



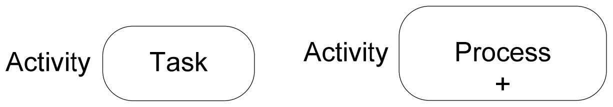
A start event starts the process flow and is described as a circle.



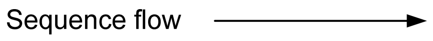
An intermediate event occurs during the course of a process flow and is described as two circles.



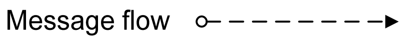
An end event stops a process flow and is described as a bold circle.



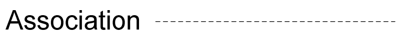
An activity is represented with a rounded-corner rectangle and shows us the kind of work which must be done. It can be a task or a sub-process. If it is a sub-process it has a plus sign in the bottom line of the rectangle.



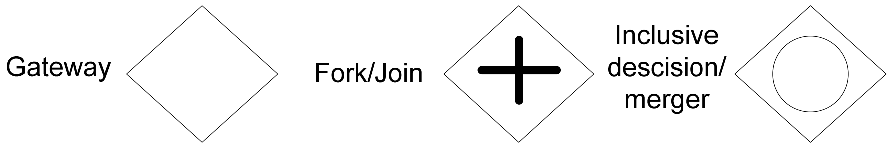
A sequence flow is represented with a solid line and arrowhead and shows in which order the activities will be performed. A diagonal slash across the line close to the origin indicates a default choice of a decision.



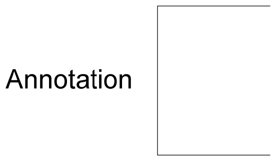
A message flow is represented with a dashed line and an open arrowhead. It tells what messages flow between two process participants.



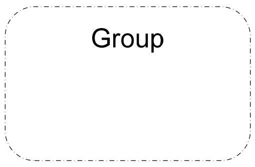
An association is represented with a dotted line and a line arrowhead. It is used to associate an artifact, data or text to a flow object.



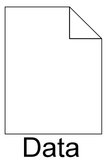
A gateway is represented with a diamond shape and will determine different decisions. It will also determine forking, merging and joining of paths.



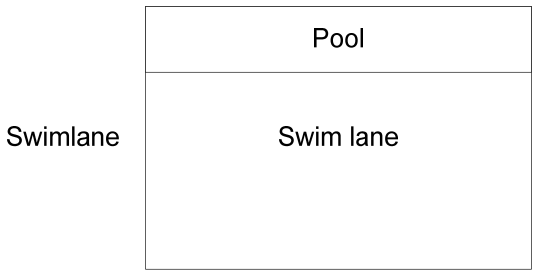
An annotation is used to give the reader of the model/diagram an understandable impression.



A group is represented with a rounded-corner rectangle and dashed lines. This symbol is used to group different activities and it does not affect the flow in the diagram.



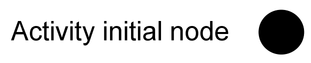
Data Objects are used to show the reader which data is required or produced in an activity



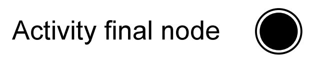
A swim lane is a visual mechanism of organizing different activities into categories of the same functionality. There are two different swim lanes, and they are:

Pool: A Pool is represented with a big rectangle which contains many flow objects, connecting objects and artifacts.
Lane: A lane is represented as a sub-part of the pool. The lanes are used to organize the flow objects, connecting objects and artifacts more precisely.

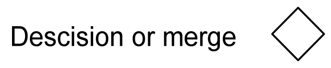
10 Appendix B



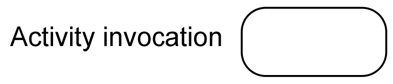
Activity initial node starts the process in the Activity Diagram and it is represented by a black circle.



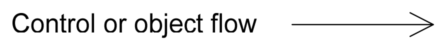
Activity final node shows where the process stops in the Activity Diagram at it is represented with a black circle with a white circle outside of the black one.



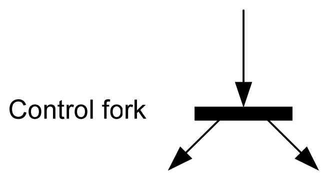
The decision or merge elements makes it possible to make a choice between different options. The different options are written next to the element.



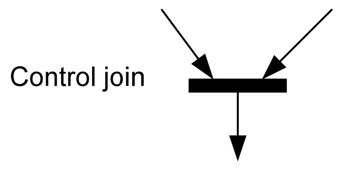
This symbol shows the activity and the name of the activity is written inside the symbol.



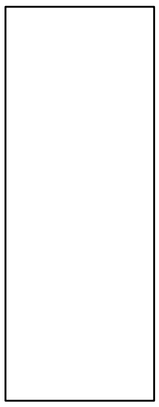
The control or object flow shows in which order the activities will be executed



The control fork element is used to show that the activities that follow can be executed parallel or in different orders.



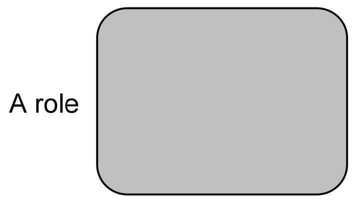
The control join element is used to show that the previous activities must be finished before they can be joined. The joining activities can be executed in whatever order but they must be finished before they can join.



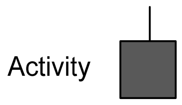
Swimlane

With a swim lane it is possible to organise an Activity Diagram so it shows that different resources are responsible for different activities. A swim lane is named after the resource responsible and then is the activities that the resource is responsible for placed inside the swim lane. For an example see Figure 12 earlier in this study.

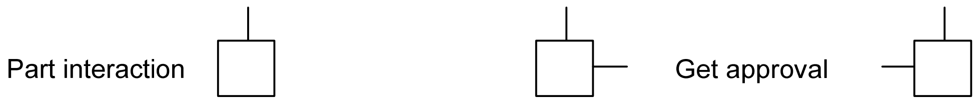
11 Appendix C



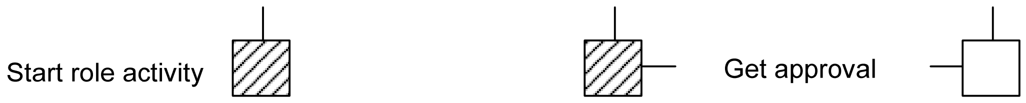
A role is a collection of activities that when they are executed together achieve a goal



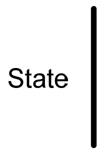
Activity is what people do and it is shown with a black box. An activity is placed inside the role that shall execute the activity.



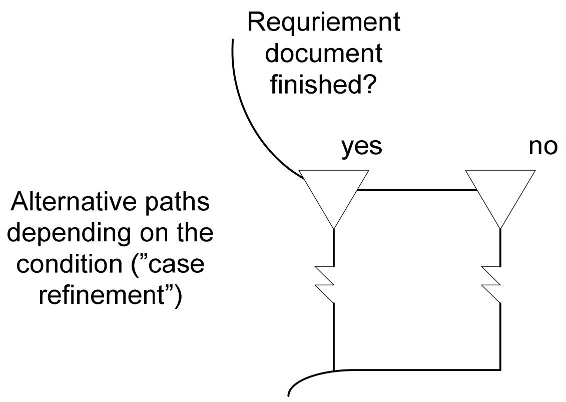
Part interaction is when different roles are interacting with each other, e.g. get an approval for something. The meaning with the interaction is written between the two parts as in the example above.



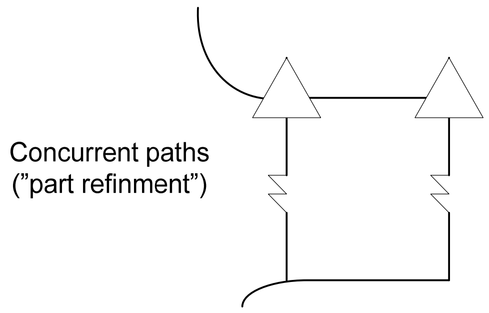
The roles that start or initiate an activity are marked with a striped box. The white box is thus the receiver.



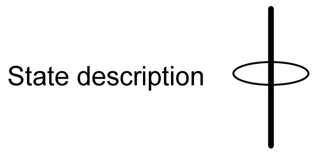
The activities are organised in states. When an activity is executed and finished it will enter the next state that for instance can be another activity. A state is described by a vertical line that connect activities.


Requriement document finished?
yes
no
Alternative paths depending on the condition (”case refinement”)

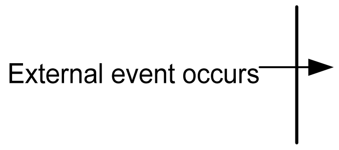
Alternative paths are leading to different paths depending on the answer for the condition stated at the start.



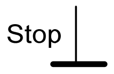
Concurrent paths can be executed parallel and the order they are executed don’t matter.



State description is used to describe the condition between two activities or to label certain states. It can also be used to label the goal with a task.



This symbol marks where a process starts or it can be used as a trigger for a new action inside a process. It can also be used as a “wait” signal.



This symbol is used to mark where a process or activity stops.

12 Appendix D

The role of described processes

I believe described processes are important for discussing improvements.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for communicating knowledge and competence.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for storing organisational knowledge.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for finding weaknesses or problems.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for measuring improvements.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for increasing product quality.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe described processes are important for other reasons (which?).

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

Attributes for Process quality

I believe it is important for high process quality that the model corresponds to “the real world”.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality with a low number of levels (clicks) to reach information.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that the symbols are easily understood.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that the flows are easily understood.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that a standard notation is used.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that common process modelling guidelines are used.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that roles are clearly shown.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that deliverables are clearly shown.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe it is important for high process quality that activities are clearly shown.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

I believe something else (what?) is important for high process quality.

☐ ☐ ☐ ☐ ☐ ☐

Strongly disagree Strongly agree

Modelling tools

What tools for modelling software processes do you know of?

I believe tools for modelling software processes are important.

☐	☐	☐	☐	☐	☐
Strongly disagree					Strongly agree

Which of the modelled processes do you prefer?

☐	☐	☐	☐
A	B	C	D