Accepted to SIG-PM 98, Linköping, 1998

Information Management in the Product Development Process

Lars Taxén
Ericsson Utvecklings AB
Box 1505, S-125 25 Älvsjö, Sweden
+46 8 727 3235
Lars.Taxen@uab.ericsson.se

Abstract

In order to succeed with the development of complex systems, it is necessary to manage the information produced throughout the development process. In this paper, we will describe a framework in which conceptual business models, entity based process models, a data model for separation of concerns, an object oriented PDM system and an information system architecture are all combined into a practical and flexible support for information management. Some examples from incremental development of software for the AXE switching system will be shown.

1. Introduction

The telecommunication market is changing very rapidly mainly due to two forces: the deregulation with the entering of many new operators leading to more competition, and the proliferation of new technology, e.g. mobile communications, intelligent networks, Internet etc. This has put a demand on the suppliers to be more reactive and flexible to the market needs, i.e. shorter lead-times and more flexibility in handling late and changing requirements. This is a very challenging change considering e.g. size, complexity and the requirements on in-service performance (up-time) of telecommunication systems.

This project has been supported by the Swedish National Board for Industrial and Technical Development, NUTEK, project P10518-1.

When developing complex systems like these, a vast amount of information is produced. To be useful, this information has to be handled in some way, i.e. it must be labelled, structured, organized etc. Some of this information is temporary and is for example used only during the design of a particular device. Other information is perhaps archived over the entire life span of a product. In this paper, we are concerned with information that is vital for controlling the development of complex systems. Since this information must be handled in a strict way we talk about *managing* the information. The items that are subject to management are called *managed items*. In short, managed items are concerned with information that is needed to successfully plan, coordinate and monitor a development project.

Managed items are usually associated with configuration management of products and documents, that is, identification, change control, status accounting and auditing. However, this is not sufficient to manage complex systems. It is also very important to consider the following aspects:

- The *context of the development*, i.e. which items are we managing in the business, and how are they related to each other? Very often, this is implicit or tacit knowledge in the organisation, but with the increasing pressure on business to change, it will become very important to know exactly what is managed.
- The *information flow* during the development, i.e. assuming that we know what items are managed, in what order are they processed, and which are the dependencies between these?
- *Separation of concerns*, i.e. what items are associated with the structuring of a complex system into parts, which can be developed more or less independently of other parts? Interfaces and specifications belong to this class of items.
- *Dependencies* in the system, i.e. how can we perform all kinds of tracing between managed items, e.g. trace customer requirements to items that implement these requirements?
- *Validation and verification* information, i.e. items which show how the system is validated ("do we build the right system?") and verified ("do we build the system right?").

A telecommunication system is usually a heterogeneous system that includes software, hardware, mechanical parts, documentation etc. Thus, many different types of support systems for information handling must be utilized during a development project. In order to clearly define the roles of these different systems, we need an *information system architecture*.

Usually, configuration management (CM) is performed as a separate process. The approach presented in this paper is to find solutions to the requirements above which include the CM activities. The suggested approach is to define static and dynamic *models* of managed items and to implement *tool support* from these models in the following way:

- The context of managed items are modelled by conceptual models in Object Modelling Technique (OMT) notation.
- The information flow impacting managed items are modelled by entity based Information Flow Diagrams [1].
- Managed items which impacts separation of concerns, dependencies, validation and verification are structured according to the Specification Based Data Model (SBDM) [2].
- The models and CM activities are implemented in Matrix, an object oriented Product Data Management (PDM) system from Matrix-One Inc.
- The information system architecture is based on structuring the information handling into activities associated with design, management and global access.

We will describe each of these solutions and how they are combined to fulfil the requirements stated above. In doing so, we will use examples from design projects at Ericsson. Finally, we will summarize the benefits from this approach.

2. Context Modelling by OMT

The context model defines the scope of the business regarding what items are managed, and how they relate to each other. This model serves several purposes. First, it is a common reference for the persons involved in the development, and secondly, it is part of the specification for information management tool support. The first aspect is usually the most difficult to achieve. Without an intersubjective understanding of the business at hand, it is very difficult to succeed with the development of complex systems. The example in figure 1 is taken from a new method package for feature based incremental development of the AXE switching software [3]. The final model was completed only after lengthy discussions and more than 30 revisions. It is now accepted as a common standard model within a large designer community at Ericsson. How this was achieved is discussed in [4].

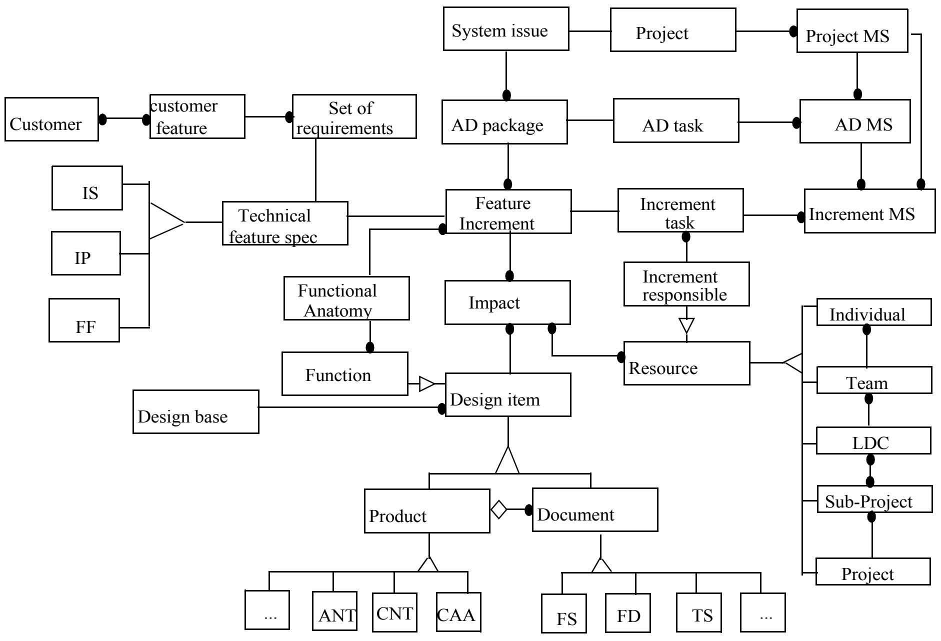


Figure 1. Conceptual model of Feature Based Incremental Development

3. Information Flow Diagrams

Information flow diagrams (IFD) show how the information flow during a development task by modeling the life cycle status of managed items and the dependencies between them. These diagrams belong to a specific class of process models called entity based models [1], which have in common that they focus on the coordination of the information produced during the process, rather than coordination of activities. Entity based models are suitable in dynamic development situations, where the growth of information is reasonably stable and activities and tools are used in an rather ad-hoc manner to produce the result. This is often the case with the development of complex systems.

An IFD concerns parts of the process which have a specified functionality, e.g. design of a printed circuit board. The model shows the managed item as horizontal lines, and the activities at the bottom. Arrows pointing down from a line indicate that this item is input to an activity. Arrows meeting a line indicate that a result has been produced, which change the status of the managed item. Horizontal arrows indicate input and output of the process.

The example in figure 2 shows the planning procedure for the feature based incremental development, corresponding to the conceptual model in figure 1. Together, the conceptual model and the IFD show the static and dynamic aspects of the items being managed.

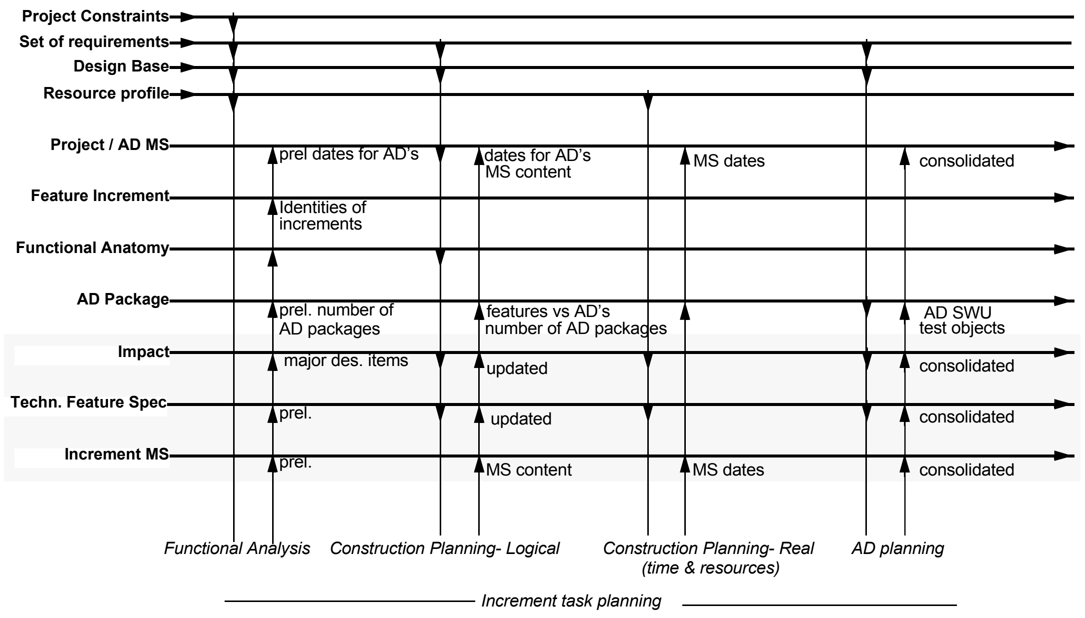


Figure 2. IFD for Feature Based Incremental Development

Another example shows the design of a multi-chip module:

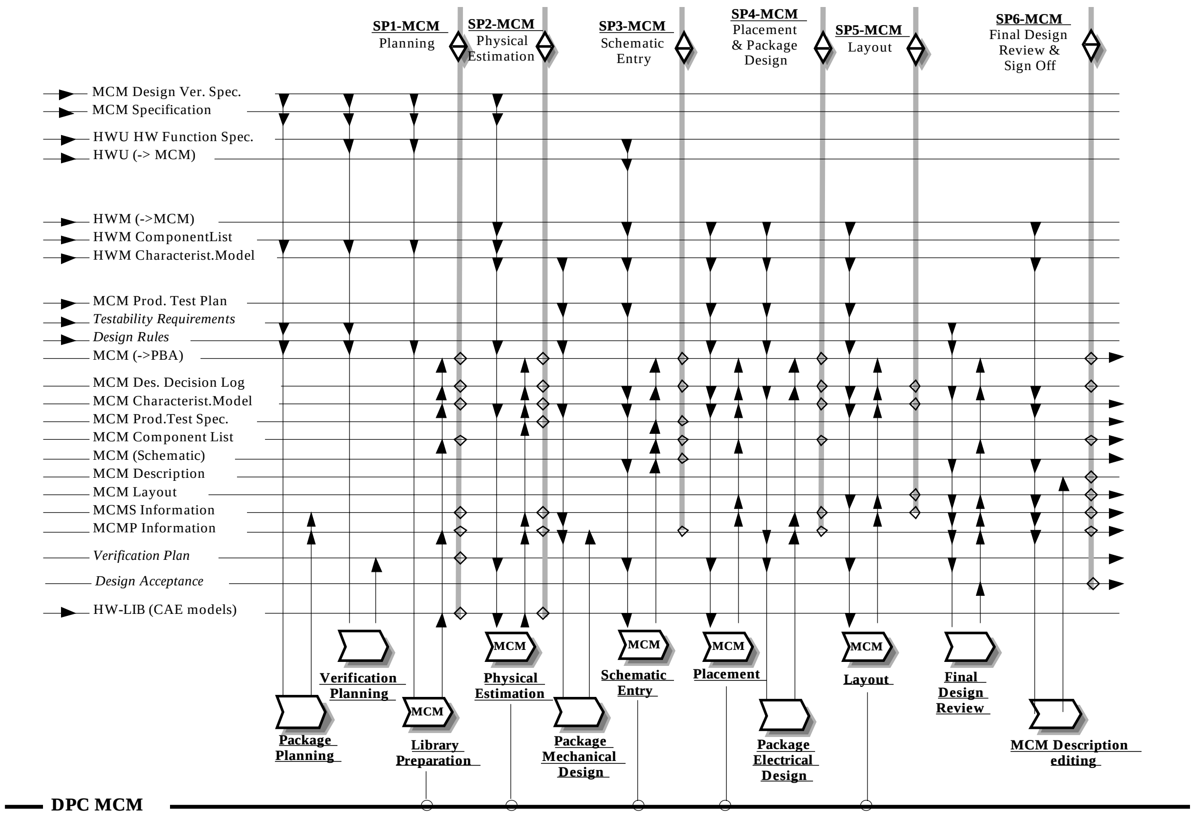


Figure 3. IFD for Multi Chip Module Design

4. Specification Based Data Model

The Specification Based Data Model (SBDM) models the system as alternating black and white boxes [2] in terms of specifications and implementations, and the dependencies between them:

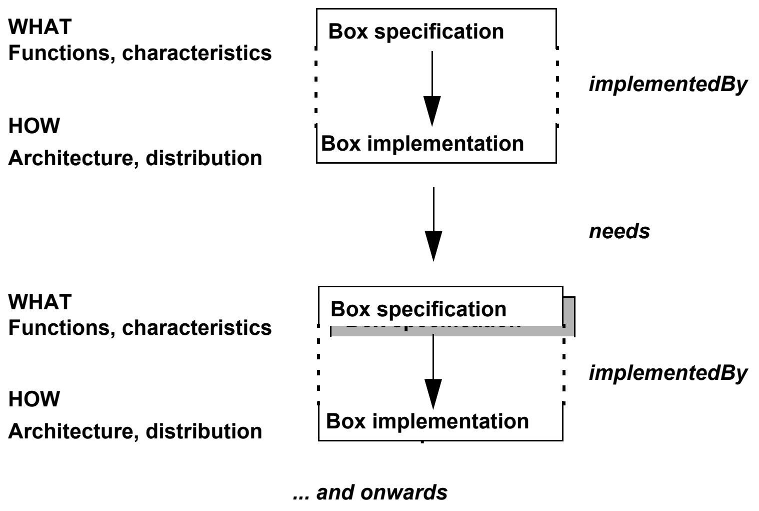


Figure 4. Specification Based Data Model

By defining the managed items *specification* and *implementation* and structure these according to SBDM, it is possible to meet the requirements regarding tracability, separation of concerns, verification and validation. For example, verification is done by comparing an implementation against its specification according to the *implementedBy* relation, while validation is done on a specification against all *needs* relations on that specification.

Specifications will sometimes have the role of a contract between development groups working with different parts of the system. Since the specifications and implementations are managed items, they can be put under change control and thus reduce the risk of uncoordinated changes.

SBMD is invariant to the implementation technology used, and thus, it is possible to structure the whole system in this way, regardless of weather the implementation is software, hardware, building practice etc. This means that tracing can be done from the specifications on the highest level of the system across the implementations and specifications on all levels down to the specifications of external parts needed for the system.

It is also possible to configure the system from this model by defining what implementation shall be included on a certain level in a configuration. In addition to this, SBDM makes it possible manage the system in a number of other aspects, like identifing what parts of the system are sourced, or defining different implementations to the same specification (for example exchanging a software implementation for a hardware implementation to improve the performance).

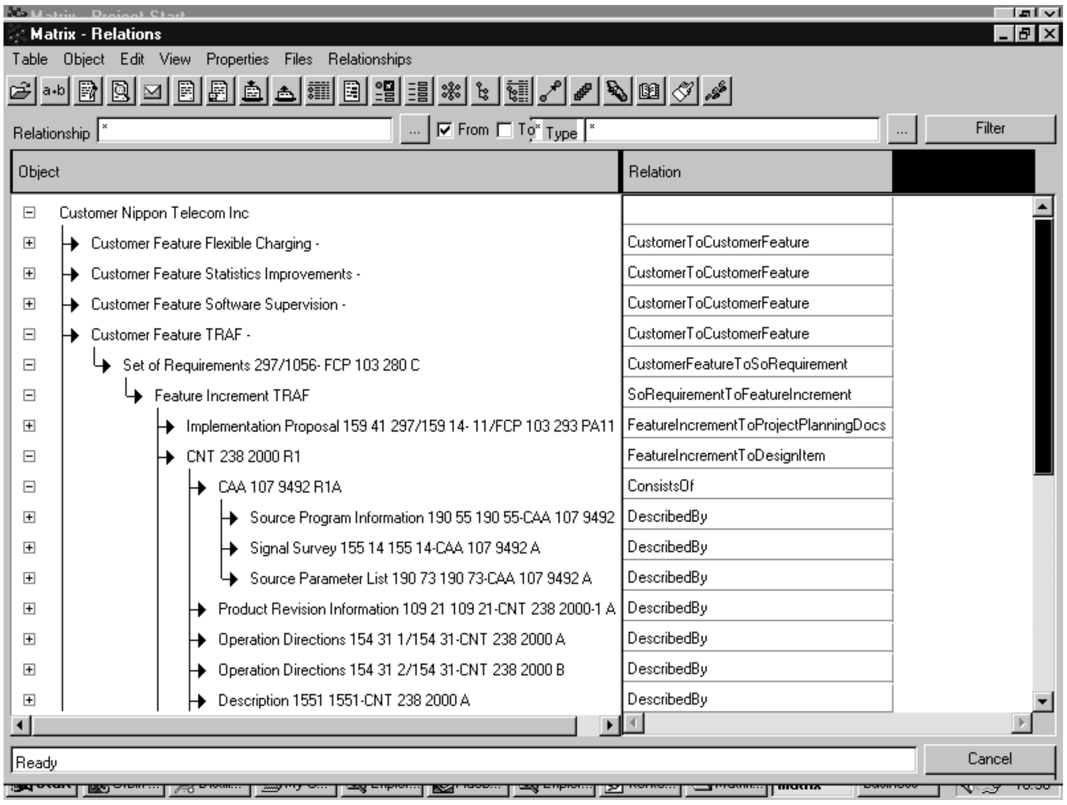
5. Matrix PDM system

Matrix is an object oriented PDM system from Matrix-One Inc. Ericsson is currently using Matrix as a prototype tool for managing feature based incremental development. The experiences from using the tool are very encouraging. The business model, i.e. the conceptual model and the information flow diagram, can be implemented in Matrix straightforwardly by defining object types, relation types and life cycle types (called policies) corresponding to these models. This is all done using the graphical user interface. Since it is very easy to modify the implementation, it is possible to start with a coarse model and then iterate between models and tool implementations. In this way, support for the models will evolve simultaneously with the models.

The standard features of Matrix for identification, version- and revision management, event logging, notification, access rights definitions etc. is utilized to define ordinary CM tasks like Change Request handling, requirement management etc.

Within Ericsson, it is also important that models can be adapted to local needs at various sites around the world. Since the modeling in Matrix is so flexible, this can be done by instant prototyping on spot, starting from a standard model. The results can be distributed immediately worldwide using the Java based web interface to Matrix. Client software can be easily installed to access a remote server, and we currently have examples of clients in Germany accessing a server in Sweden.

An example from the feature based incremental development is found in figure 5, which shows how managed items can be traced from customer features down to single documents impacting the implementation of that feature. It can also be noted that the managed items and the relations are exact instances of the types in the conceptual model described in chapter 2.



Object	Relation
Customer Nippon Telecom Inc	
Customer Feature Flexible Charging -	CustomerToCustomerFeature
Customer Feature Statistics Improvements -	CustomerToCustomerFeature
Customer Feature Software Supervision -	CustomerToCustomerFeature
Customer Feature TRAF -	CustomerToCustomerFeature
Set of Requirements 297/1056- FCP 103 280 C	CustomerFeatureToSoRequirement
Feature Increment TRAF	SoRequirementToFeatureIncrement
Implementation Proposal 159 41 297/159 14- 11/FCP 103 293 PA11	FeatureIncrementToProjectPlanningDocs
CNT 238 2000 R1	FeatureIncrementToDesignItem
CAA 107 9492 R1A	ConsistsOf
Source Program Information 190 55 190 55-CAA 107 9492	DescribedBy
Signal Survey 155 14 155 14-CAA 107 9492 A	DescribedBy
Source Parameter List 190 73 190 73-CAA 107 9492 A	DescribedBy
Product Revision Information 109 21 109 21-CNT 238 2000-1 A	DescribedBy
Operation Directions 154 31 1/154 31-CNT 238 2000 A	DescribedBy
Operation Directions 154 31 2/154 31-CNT 238 2000 B	DescribedBy
Description 1551 1551-CNT 238 2000 A	DescribedBy

Figure 5. Tracing requirements to implementation documents

6. Information handling System Architecture

The information handling system architecture is based on the grouping of handling functions into three categories:

- *Design environment (DE)*, where technology specific design is done.
- *Management environment (ME)*, which comprises operations on managed items as described in this paper.
- *Common environment (CE)*, which concerns global access and archiving.

The ensemble of these functions can be regarded as specifications in the SBDM sense. The systems implementing the DE functionality are usually commodity systems that are sourced and used more or less without modification. The systems in CE implement worldwide core business functionality, and have usually been used for a long time. The DE- and CE systems are less affected by changes in the company business. However, for system used in ME, it is very important that the business model can be easily implemented and changed, since these systems will be directly impacted by customer changes, adaption to local circumstances, business restructuring etc.

The architecture can be illustrated as in figure 6, which show the functionality grouping. For each environment, it is shown some Ericsson implementations for various areas like software design, hardware design etc.

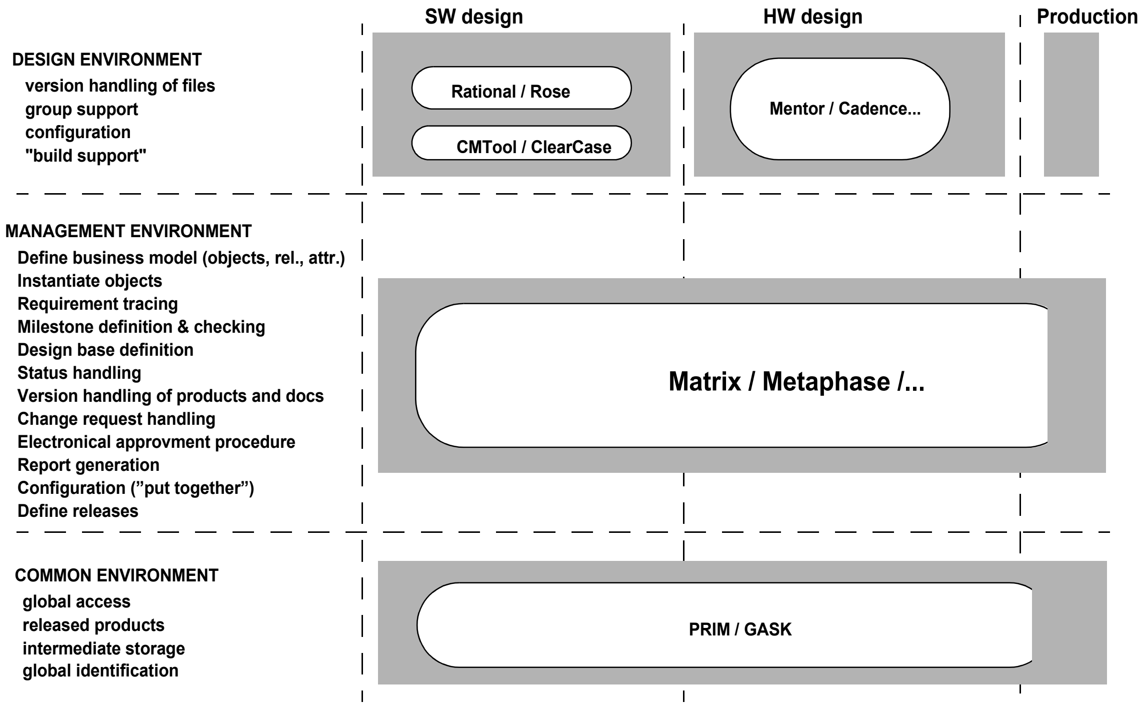


Figure 6. Information system architecture

This architecture makes it possible to discuss the role of each system, and also what kind of information must be exchanged between the systems. The architecture is also used when evaluating the properties of systems that implement the same functionality.

7. Summary

The solutions to the information management problem presented above are based on practical experiences from the development of telecommunication systems. The models and tools can easily be changed and adapted to address specific needs. Since the solutions are general and not geared to the specific needs of Ericsson (besides some the information handling systems), the framework should be applicable for managing other types of system developments as well.

Furthermore, the solutions can be utilized for creating, deploying and maintaining new knowledge within an organisation. Since this concerns assets which are not tangible, this task is very difficult and often grossly underestimated. This is especially true for a large company like Ericsson, which has e.g. more than 10 000 software designers worldwide working with a variety of applications in many countries. The graphical models can be discussed and agreed upon by the project members. Also, since the models can be implemented straightforwardly in Matrix, it is possible to directly try out the ideas that are generated during the model discussions. In this way, an inter-subjective understanding about the task will emerge between the persons involved, concurrently with tool support for the task. Thus, the new subjective knowledge gained by the members will rapidly be objectified into concrete company assets in the form of models and tool support, which can then be passed on to other employees in the company. This process of establishing a new knowledge in an organization, has been treated in more detail elsewhere [4].

References

1 Humphrey, Watts S. and Kellner, Marc I. (1989): *Software Process Modeling: Principles of Entity Process Models*. SEI-89-TR-002

2 Gandhi, M. and Robertson, E. L (1995): *SDBM as a Model for Codesign Data*", in "Computer Aided Software/Hardware Engineering," IEEE Press, Piscataway.

3 Taxen, L. and Karlsson, E. A (1998): *Incremental Development for AXE 10,* Frankfurt, Ericsson Conference on Software Engineering.

4 Taxen, L (1998): *Organisatorisk kunskapsförändring - en fallstudie,* Linköping: uppsats i doktorandkurs Kunskap och Handling (in swedish).