Ontology Guided Evolution of Complex Embedded Systems Projects in the Direction of MDA

Lars Pareto[1], Miroslaw Staron[1], and Peter Eriksson[2]

[1] IT University of Göteborg, 412 96 Göteborg, Sweden
{lars.pareto,miroslaw.staron}@ituniv.se
[2] Ericsson Software Research, Ericsson AB, Sweden
peter.r.eriksson@ericsson.com

Abstract. Implementation of MDA in large, product developing organizations involves changing processes, practices, tools, and communication infrastructures. The paper presents a case study, in which modeling related needs of a unit within Ericsson were compared to features of current and envisioned MDA tools, using qualitative methods. The paper's main contribution is an ontology defining areas and sub-areas of improvement associated with the introduction of MDA in complex embedded systems projects. The ontology is grounded in interviews with senior modellers at Ericsson and in survey publications from within the field of MDA. It identifies 26 improvement areas concerned with model content, modeling activities, and the management of modeling projects. The ontology has been presented to stakeholders within the unit studied, with positive feedback: appreciated were its groundedness, traceability, holistic scope, and potential as platform and checklist for several recurrent analysis and communication tasks related to software process improvement within Ericsson.

1 Introduction

To implement MDA in a large organization with products on the market is, in many senses, a wicked problem [1]: required changes are plentiful and interrelated; data on which to base estimates of costs, benefits and risks are scarce; the implementation target is moving. Yet, many companies realize that their future software development requires better utilization of modeling technologies—they are just unsure about the path.

The purpose of this paper is to support organisations in this situation—large, product developing companies that strive to increase their use of modeling in the direction of the MDA vision, but whose decision making regarding this stutter because of too many risks and constraints.

The paper approaches this problem from the perspective of practicing software engineers in commercial, complex embedded systems projects, already using UML for informal modeling. Our view of such projects is captured by the conceptual framework in Fig. 1: we view software as produced by engineers in specialized roles (Requirement engineer, etc.), who are steered by processes (defining who should produce what when for whom), and who communicate under the constraints of a communications infrastructure (consisting of model repositories and tools); the project is steered by business goals (such as reducing the time to market for new products) and subject

K. Czarnecki et al. (Eds.): MoDELS 2008, LNCS 5301, pp. 874–888, 2008.

to business constraints (such as bounds on development costs). We view project improvement as the matter of engineering features of internal processes and infrastructure towards several requirement sources (needs of every role, business goals, and business constraints), utilizing features of modeling technologies and process frameworks developed outside the project. By complex embedded system we mean a large, special purpose, real-time, multiple processor, computer system that is part of a larger, technical system.

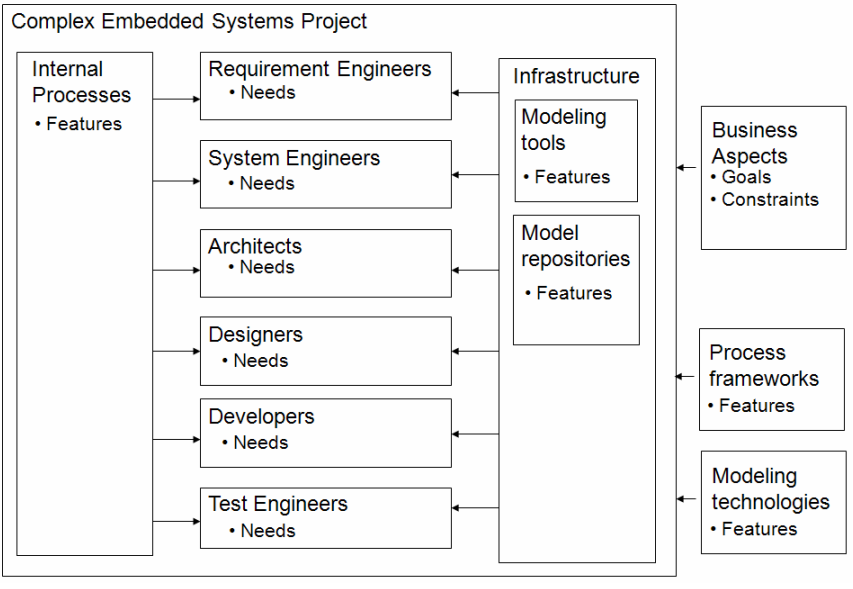

Fig. 1. Improvement variables in complex embedded systems projects

The research question addressed in this paper is which improvement areas projects of this kind face when implementing MDA. Put precisely: which areas of improvement do requirement engineers, system engineers, architects, designers, developers, and test engineers, developing and maintaining complex embedded systems, need to be concerned with when proceeding from informal UML-based software development to MDA? By MDA, we mean the use of domain specific UML dialects and model transformations for specification and realization of software. (Stahl [2] gives an overview.)

The paper addresses this question by an exploratory, holistic, single case, case study [3]. The unit of study is a subproject within Ericsson developing embedded software for a constituent part of a mobile-communications-network product. The main outcome of the study is an ontology defining improvement areas associated with the introduction of MDA in complex embedded systems projects.

The paper is organised as follows: we describe our research design (Sec. 2), our case (Sec. 3), our study of this case (Sec. 4–5), the ontology resulting from this study (Sec. 6), and the use of it in process evolution (Sec. 7); we discuss limitations of our study (Sec. 8), and related work (Sec. 9); finally, we summarize or findings and draw conclusions about the approach (Sec. 10).

2 Research Design

Our epistemological position is interpretative research [4]; our research strategy is qualitative research [5]; our data analysis method is grounded theory in the tradition of Strauss [6]. The research design is outline in Fig. 2. Data collection has proceeded by *semi-structured interviews* and selection of written sources from within the modeling research community. Data analysis techniques used are *open coding* (conceptualization of data sources using descriptive codes), *categorization* (grouping of codes with commonalities into categories), and *axial coding* (relating categories to subcategories). Our application of these techniques has been guided by *technology roadmapping* [7] (that emphasizes the modeling of needs and technological options in a common framework). The analysis outcome is a simple, informal ontology with inclusion hierarchy [8] (also known as a taxonomy) characterizing areas and sub-areas of improvement associated with the introduction of MDA in complex embedded systems projects. Analytic generalization [3] has been used to obtain an ontology presumably useful for complex embedded systems projects in general. The ontology's validity relies on the grounded approach and feedback from practitioners.

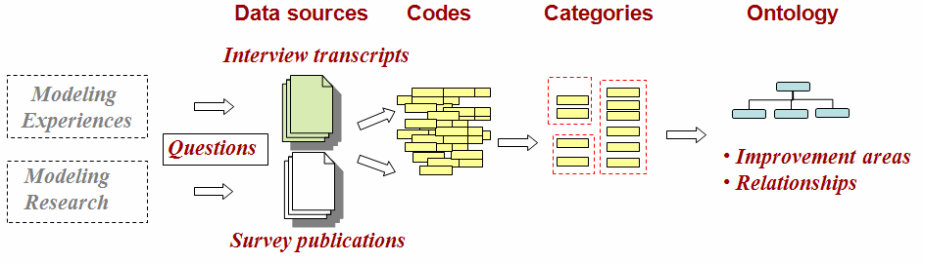

Fig. 2. Research design

Research has been collaborative [9], and involved three Ericsson insiders (a senior technical specialist, a software architect, and a project manager), and two outsiders (one software engineering researcher with a background in stereotype-based language customization, and another with a background in software quality and programming language semantics). Insiders have facilitated the study, defined the problem, set the scope, selected informants, and given recurring feedback on the study's development; outsiders have designed the study, conducted data collection and analysis, framed the study, and communicated the results.

Our choice of a single case case-study is partly due to the revelatory nature [3] of our research question (we seek to elicit areas of software process improvement in a certain situation, rather to confirm or refute an hypothesis), and partly due to our use of enquiry based qualitative research (which is limited to small samples): distributing available resources over the study of several cases had been at the cost of penetration of individual level needs and inconsistent with our choice of method.

The particular unit was chosen because the Ericsson insiders were familiar with, and had a direct stake in improving this unit: this improves the quality of sampling, the data analysis, and eases the validation results (compared to the choice of some unfamiliar unit).

3 The Case and Its Context

Ericsson has a long tradition in model driven development: the use of software models with associated semantics dates back to 1967, with the AKE switch, first delivered in 1971, built using such [10]. Ericsson contributed to development of SDL [11] during the 70s, and applied SDL extensively during the 80s; use case based modeling was pioneered by Ericsson in the 80s [12]; in the 90s, Ericsson was an early adopter of RUP [13]; today, MDD plays a central role in several Ericsson product lines, and model based software engineering is recognized as a prioritized area of improvement.

The project, to which the unit belongs, uses UML for requirements modelling, system design, systems architecture work, software architecture-work, detailed software design, and software implementation. Other notations are also used: requirement engineers also use textual use cases and supplementary requirements specifications; hardware designers use block diagrams, and Mealy/Moore state machines (of their trade). Where UML is used, it is often complemented with text based notations, e.g., for signal and protocol specification, and with informal text-based specification documents for aspects not easily expressed in standard UML, e.g., non-functional requirements or configurations.

The unit operates in the following technical context: it develops the software part of a subsystem; the software runs on in-house developed, multi-processor hardware (involving digital signal processors, field programmable gate arrays, and ordinary processors); it is subject to real-time constraints (response time, throughput, and space bounds), compatibility constraints (RTOSes and in-house developed platforms), and special run-time requirements (monitoring, configuration, upgrading, and rollbacks).

The project has a conventional line organisation. The unit itself has approximately 100 engineers situated on a single location; the whole project is much larger and distributed over several locations. The unit is divided into six sub-units: two responsible for software specification, three for software implementation and maintenance, and one for integration and validation.

4 The Unit's Needs

4.1 Interviews with Engineers

Informants were selected by the insiders, using the following criteria: the scope of their experience should be wide; they should have worked with model driven development in their daily work, they should understand model driven development from the perspective of several roles (and those of architects and designers in particular).

The enquiries were semi-structured interviews revolving around the following set of questions:

> What, do you think, Ericsson hopes to achieve by model driven development? Do you believe in this for Ericsson as a whole / for your project / for your role? What improvements of your project do you spontaneously associate with model driven developments? Which deteriorations do you associate? Is there some slave-work / double work, do you think, that could be automated/eliminated? Do you see

> any obviously inefficient practices that ought to be improvable? What's your view on the use of modelling for the activities in this list: requirements work, architecture work, detailed design, estimation, function testing, subsystem testing, documentation work, maintenance, code generation, configuration/run-time-use, change request handling, defect handling? How could modelling be used in the near future? Do you have any vision for your own, work/for the work of others? Are you aware of any modelling success stories inside or outside Ericsson?

Questions were designed to, in an non-leading way, bring out the personal attitude and perceived goals of model driven development, to trigger an open-ended exploration of the informants perceived own needs, those of others, and company objectives. To avoid discussions on the differences between MDA and model driven development, questions purposely referred to the latter. The need for MDA was probed for indirectly, through questions to reveal a need for automation.

Interviews lasted 1-2h each, and the resulting transcriptions amounted to 60 pages of 10 pt, singly spaced text files. A summary of the interviews is given in Table 1. In addition to the interview transcripts (S_1,...,S_4), notes from a preparatory meeting, in which four designers discussed what to bring forward in an upcoming meeting with a tool vendor, was added as a supplementary data sources for needs (S_5).

4.2 Interpretation of Interviews

Interview transcripts were coded in search for *needs*—a concept prevalent in technology roadmapping—in a broad sense: we included *direct needs* expressed by the informants themselves and *indirect needs* inferred from their descriptions of situations or problems; in addition to *unsatisfied needs,* we included *satisfied needs*, not to be overlooked in process change; we included *realistic needs* whose satisfaction seemed plausible as well as *wishful needs* whose satisfaction seemed to require advances beyond state of the art. The scope of the identified needs was model based software engineering and management, which is a larger scope than MDA technologies, but necessary to consider when implementing MDA.

To make the set of needs intellectually manageable and to facilitate communication of our results to non-analysts within Ericsson, interpretation was subject to the following principles, which emerged during the interpretation: each need should be (1) abstract enough to fit on a single line, but (2) concrete enough to suggest a specific improvement (or a set of specific improvements); (3) needs shall be *distinct*, by which we mean that all passages in the data sources referring to the same phenomenon are represented by a single need; (4) unless 2 is violated, similar needs should be coalesced into one more abstract needs; (5) needs that apply to several roles should be generalized into such.

> We illustrate the interpretation process and these principles through the need
> Subsystem level cohesion analysis

which is the analysts interpretation of the following two passages of text (in their contexts):

"It's like, should I introduce a new component, or should I put the function into this old one"	"we really tried to understand the problem [...] to avoid the solution from being too scattered"

This need is indirect: from a particular work situation, the analyst has recognized that engineers are concerned with component cohesion, concluded that analytic tools for reasoning about cohesion would be helpful, and phrased this as a need; later, the phenomenon has reappeared in a slightly different work context, similar enough to be an instance of the same need (possibly along with other needs). The need is unsatisfied: the unit's engineers do not use tools for cohesion analysis in their daily work. The need is realistic: several theories and tools for cohesion analysis of models are available.

Further, the need satisfies principles 1–5: it is short, it points into a concrete domain, i.e., model quality metrics, it represents several instances of the phenomenon in the text, and it entails needs of architects and designers. The need could be further improved with respect to principle 5 by removing the restriction to subsystem level, as cohesion analysis is a likely concern at the system and implementation levels too.

In all, 269 distinct needs were identified in 36254 words originating from 7 engineers in 3 roles, at system (S) and subsystem (SS) level. The number of inferred needs and examples of needs for each data source are given in Table 1. (Some needs were mentioned by several informants, and one informant represented in two data sources, which is why summing columns 3 and 5 results in larger numbers than those above.)

Table 1. Some of the Unit's needs and the underlying data sources

S_1	May 14, 2007	Architect (SS)	14 588 words; 20 pages	143 needs
Subsystem design- and implementation models shall be distinct. E-sketching using UML. UML for use case analysis. UML for reverse engineering. UML for refactoring. OO framework design supported by workflow. Deployment modeling. Implementation modeling should be optional. …				
S_2	May 16, 2007	System Eng (SS)	8 297 words; 11 pages	67 needs
Open formats for models. Tool integration flexible. Vendor independence. Documentation globally searchable. Documentation should be structured for large information volumes. Baseline handling for reading users should be simple. Requirements tracing from model. Model oriented description at system level. Text search into model database should be global. ...				
S_3	May 16, 2007	Developer (SS)	11 994 words; 17 pages	85 needs
Subsystem level analysis modelling. Subsystem level cohesion analysis. Subsystem level architecture knowledge among designers. Analysis modeling and implementation modeling shall be distinct. Education in work-task specific modelling. Clear separation between specification and white box interface synchronization.				
S_4	July 7, 2007	Architect (S,SS)	845 words; 2 pages	18 needs
Better definition which elements are in diagrams. Overview mechanisms for complex models. Links between information in legacy elements and system model. Improved inspections of models. Guidelines for managing documentation update. Auto-generation of documents from design models. Guidelines for model walkthroughs. Specifications written at the same level of abstraction. Single point of adding information in model. System model is the primary source of information. ...				
S_5	April 27, 2007	1 Sys., 3 Dev. (SS)	510 words, 8 pages	22 Needs
Better definition which elements are in diagrams. Overview mechanisms for complex models. Links between information in legacy elements and system model. Improved inspections of models. Guidelines for managing documentation update. Auto-generation of documents from design models. Guidelines for model walkthroughs. Specifications written at the same level of abstraction. Single point of adding information in model. System model is the primary source of information. Requirements modeling using deployment diagrams. System model is the primary source of information. Align legacy documents and new tooling. ...				

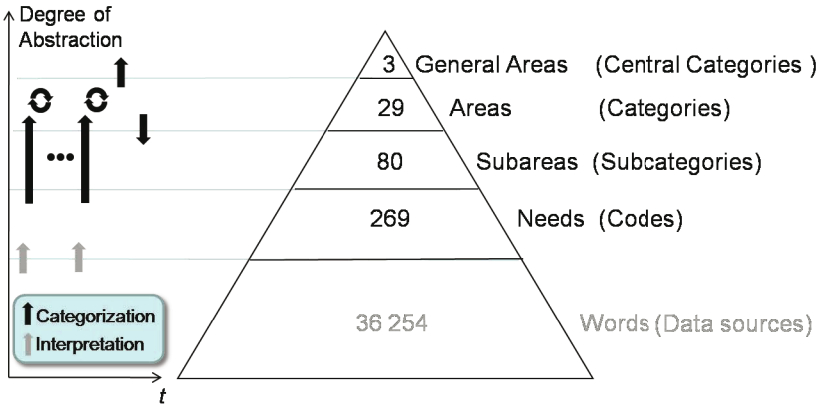

Fig. 3. Categorization process (left) and outcome (right)

4.3 Categorization of Needs

The categorization of the needs was iterative, incremental, and interleaved with the interpretation of needs and the categorization of features (left part of Fig. 3): codes resulting from an initial interpretation of the transcripts (leftmost grey arrow) were grouped into areas of improvement (leftmost black arrow) and named; categories were then restructured and renamed (leftmost circulation-symbol) for consistency with those emerging from the analysis of the survey publication, which was conducted in parallel; categorisation and restructuring were repeated several times, when new data sources were interpreted or old data sources re-interpreted.

To make the category system suited for process improvement work (i.e., comprehensible, credible, maintainable, and usable) categorization was subject to the following principles: (1) categories should have concrete, suggestive names capturing the underlying phenomena (rather than abstract names capturing too much); (2) categories should be given meaning by characterizing definitions along with traceable connection to the underlying data sources; (3) categories should be coarse enough to allow quick classification of needs; (4) within categories, sub-categories should be used to group elements with closer relationships to each other; (5) categories should be general enough to serve as containers for both needs and features (thereby making needs and features easier to compare); (6) deep-hierarchies should be used sparingly (as an overuse makes category system difficult to comprehend and maintain); (7) categories should be role centric (to make the responsibility for satisfying needs more clear).

The categorization process involved (in addition to the grouping of needs and introduction of names and characterizing definitions) coalescing, subdividing, widening, narrowing, and renaming categories; it involved re-categorization and renaming of codes to better realize interpretation and categorization principles.

The process eventually resulted in the ontology outlined in the right part of Fig. 3 (and which is further described in Section 6). Improvement areas are found at four levels of abstraction: 3 general areas, 26 areas, 80 subareas, and 269 needs.

5 Features of MDA Technologies and Processes

5.1 Literature Search

The literature study addressed the following main question:

Which are the features of current and envisioned MDA processes and tools?

Sampling was restricted to survey publications; these were selected to cover both technical and managerial aspects from the perspectives of modeling researchers, modellers in industry, and suppliers of modeling tools. An overview of the publications used in our analysis is given in Table 2.

5.2 Interpretation of Survey Publications

Interpretation of survey publications used the following principles: (1-5) principles similar to the five used in the interpretation of transcripts; (6) process features with no other distinct phenomena than the use of a technology feature, should be implicitly defined by the latter; (7) process and technology features should not be kept apart. We motivate 6 by the following example of two possible features:

"Automated model metrics is used" (Process feature)
"Automated model metrics" (Technology feature)

Any technology feature has an obvious corresponding process feature, whose presence would clutter the ontology through redundance. We motivate 7 by the following process feature:

"Software product lines" (Process feature and technology feature)

This feature comes with certain commitments to both process (configuration is done by compilers rather than people) and technology (feature diagrams, connected to model to model transformations, and composition infrastructure in certain ways), thus it is a compact carrier of both concerns. Encouraging analysts to view all codes from both angles gives a more compact representation better coverage, and quicker classification compared to keeping process and technology features apart.

Interpretation of the survey publications, along these principles, resulted in 214 features, some of which are given in Table 2.

Table 2. Process- and technology features and the underlying data sources

S_6	MDD practices within Mototorola [14]	39 features
Data reuse between design and testing activities. Testing by co-simulation. Automatic test generation. Automatic code generation. Standardized and non-proprietary modeling languages....		
S_7	MDD research roadmap [15]	61 features
Models describe the system at multiple levels of abstraction. Formal semantics. Generation of configuration files. Synchronization transformation (Model-Code). Verification by simulation modelling. ...		
$S_{8\text{-}10}$	Modelware Metrics, Projects, Frameworks [16-21]	69 features
Maturity levels definition. Models used for production of documentation. Platform independent and platform specific models separate. Generation of implementation infrastructure....		
S_{11}	MDD technologies and tools [2]	30 features
Aspect models. Bidirectional transformation. Code and document generation automated. Config. by feature models. Domain specific modeling. M2C weaving. ...		

5.3 Categorization of Features

Categorization of features followed the same steps as, and was interleaved with, the categorization of needs. Initially, separate category system were maintained, but during the course of analysis the two were merged, for the following reasons: there was a large overlap in concepts; a common system eliminated the task of relating the two system and that of keeping them consistent; viewing needs and feature as members of the same category had analytic power (it enabled the analyses describe in Section 7.2, and simplified identification of sub-categories).

6 Ontology of MDA Implementation Improvement Areas

6.1 General Areas and Areas

The ontology is a hierarchy, in which general areas and areas form a tree, but in which sub-areas crosscut (technically a graph). Each node consists of a *name* and *characterizing definition* and is associated with a subset of all features and needs. There is a strict inclusion order: any need or feature that belongs to a sub-area also belongs to the ancestors of this node. The topmost two levels of the ontology, the general areas and areas, are given in Table 3 on next page: the three general areas

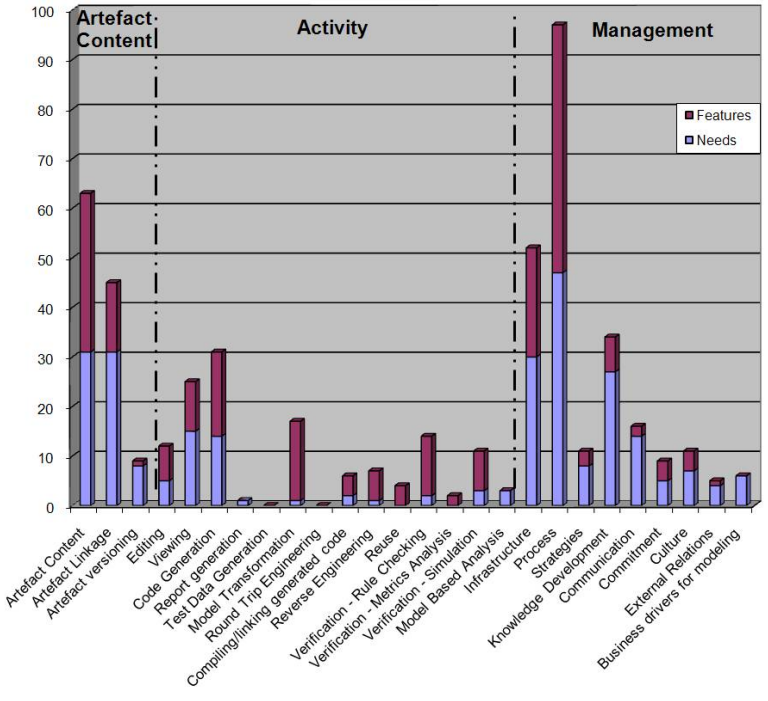

Fig. 4. Histogram showing the distribution of the needs and features over modeling areas

(Content, Activity, Management) are subdivided into 26 areas (Artefact content, etc.) each with a characterizing definition of the areas *concern*. The number of needs and features associated with each of these areas are given by the histogram in Fig. 4: the area *Artefact Content* contains 31 needs and 32 features, etc.

Table 3. Improvement areas when implementing MDA in complex embedded systems projects

Content *Information in models*	**Activity** *Operations on models*	**Management** *Socio-technical aspects of modeling*
Artefact content *The syntax and semantics of the artefact kinds, and their use.*	**Editing** *Reading, writing, modifying models.*	**Infrastructure** *Servers and tools for producing using/sharing/reusing/distributing/archiving models/artifacts and the integration of these.*
Artefact linkage *How artefacts (models, code, interface files, scripts, metamodels, ...) at different levels of modeling (requirements level, system level, subsystem level, implementation level) in different activities (specification, implementation, testing, documentation) should be linked.*	**Viewing** *Navigating and searching own models and those of others.*	**Process** *Who should produce what model when for whom.*
Artefact versioning *Information related to the evolution of models.*	**Code generation** *Automatic production of programming code in text based languages from models.*	**Strategies** *Tactics for introducing, executing, and optimizing the use of modeling.*
	Report Generation *Automatic production of documentation, and specifications from models.*	**Knowledge development** *Education (in tools, practices, abstract thinking) and internal knowledge transfer (of technologies, architectural principles, design rules).*
	Test Data Generation *Automatic generation of test scripts or test data from models.*	**Communication** *Exchange of information between roles.*
	Model Transformation *Automatic transformation of models from one kind or use to another kind or use.*	**Commitment** *Managers and engineers engagement in the introduction and improvement of modeling.*
	Round Trip Engineering *Co-existing manual development of models and code.*	**Culture** *Relative values of artefacts, roles, and tasks among managers and engineers.*
	Compiling/linking/tracing code *Integration of models with their target system incarnations.*	**External Relations** *The technology suppliers' responsiveness to organization's needs; negotiation position.*
	Reverse Engineering *Creation of model from source code.*	**Business drivers** *Economical incentives for using model driven development instead of something else.*
	Reuse *Libraries of models and patterns.*	
	Verification - rule checking *Well-formedness of models wrt. rules.*	
	Verification - metrics analysis *Computation of model metrics.*	
	Verification – simulation *Off-line execution of models.*	
	Model based analysis *Use of models to reason about the system built or the project building it.*	

Subareas
The third level of the ontology, the subareas, is given in Table 4 for the general area Content:

Table 4. Improvement subareas for the artefact-content, -linkage, and -versioning areas

Content		
Artefact Content	**Artefact Linkage**	**Artefact Versioning**
Artefact Kinds *The type of diagrams used (class-, sequence-, timing-, etc.).*	**Connectivity/ integration** *How diagrams are attached to each other by the infrastructure.*	**Conflict avoidance/resol.** *Detecting and handling concurrent changes in models.*
Abstraction *The levels of abstraction at which the diagrams are used; whether abstraction is respected.*	**Separation of Concerns** *The degree to which aspect views are distinguished in models.*	**Change book-keeping** *Versioning information, e.g., tags, branches, log appears.*
Annotation *Information not intrinsically part of the diagram (e.g., links and author tags).*	**Consistency** *Syntactic consistency across diagrams with respect to names and structure.*	**Change impact analysis** *Detecting which artefacts in which branches would be affected by a change to one.*
Modeling Areas *What are diagrams are used for (e.g., code generation, design).*	**Principality** *The recognition of some linked artefacts as authorative.*	**Baseline handling** *Annotation of artefacts as belonging to baselines, and retrieval of such.*
Modeling Tasks *Specific tasks that require specific content.*	**Pollution reduction** *Stopping lower level concepts from leaking into higher level models.*	**Access** *Ease with which artefacts may be obtained from own repositories and those of other groups.*
Artefact Semantics *Precision in and variability of meaning.*	**Propagation** *Changes to one model automatically causing updates in other.*	**Branch & Merge** *Forking, synchronizing tracks; bringing tracks together.*
Conventions *The uphold of good common modeling practices.*	**Isolation** *The ability to restructure two linked models at one end only.*	**Volatility** *The recognition of some artefacts as temporary throwaways.*
Customization *Degree to which diagram-types can be adapted to company specific needs.*	**Model-Code Interfacing** *Embedding of models in code, and the embedding of code in models.*	**Reuse** *The distinction of artefacts as general/special, shared/project-specific and stable/changeful.*
Degree of modelling *Degree to which modeling is put to use in a certain area (informal, formal, executable, complete, incomplete).*	**Access** *Ease with which artefacts developed by other groups may be obtained /updated.*	**Granularity** *Scope of syntactic units that are versioned (Model tree, Diagram, Single Transition).*
	Cohesion *The degree to which related parts are held together.*	**Isolation** *Repository structure constraining the artefact structure.*
	Linkage visibility *Whether, how, and where links appear in user interfaces.*	

7 Ontology Guided Evolution

Our main use of the ontology has been in the context of technology roadmapping.

7.1 Technology Roadmapping

Technology roadmapping [7] is a technique widely used in industry for technology strategy work. By a combination of analytical and collaborative activities, a time-based chart comprising a number of layers representing both commercial and technological perspectives—a *roadmap*—is built and maintained. Our exploration of technology roadmapping focussed on its analytical activities and relied on the ontology in the following ways.

A roadmap was obtained by (1) categorization of needs and features along the temporal dimensions in Table 5, and (2) by presenting needs and features in a matrix display [22] with categories of Table 3 as rows, and those of Table 5 as columns.

Table 5. Temporal Categories used in Roadmapping

Done *Needs already satisfied within unit.*	**Piloted/planned** *Needs soon to be satisfied.*	**Vision within WoW** *Needs whose solutions are compatible with present WoW.*	**Vision beyond WoW** *Needs whose solutions require major changes of present ways of working.*
67 needs	34 needs	107 needs	59 needs
Mature/straightforward *Features available in shelfware / books or whose implementation is standard.*	**Emerging** *Features of technologies / processes with early adopters in ind.*	**Researched** *Features of technologies / processes that have gained use in the research community.*	**Vision** *Features that are visions even to researchers.*
100 features	26 features	66 features	34 features

7.2 Ontology Based Analyses

We used the ontology in two analysis tasks associated with roadmapping: establishment of linkages [7] between needs and features, i.e., a relation defining which needs are are supported by which features; identification of gaps [7] in sets of features and needs, i.e., missing needs and features that would be present in an ideal roadmap.

Both linkages and gaps were established by crosswise comparison of needs and features and incremental definition of a relation x isSupportedBy y, stating, for example, that "feature selection tool support" isSupportedBy "software product lines". *Linkages* are the needs-feature pairs $\langle x,y \rangle$ associated by this relation. *Gaps in features* manifest themselves as needs not linked to a feature, e.g., in our set of needs and features, the need "fine grained version control" is not linked to a supporting feature, which points out a potential gap in the feature set. *Gaps in needs* manifest themselves as features not linked to a need, e.g., the feature "instance modelling of signals", which has proved valuable in contexts similar to that of the unit, did not support any need, thus pointed out a potential gap in the set of needs.

Notice that what this detects are *potential gaps* in needs: it may well be that a feature is not needed; features solving a certain need may not have been invented yet. To turn potential gaps into actual ones is always a matter of additional data collection and interpretation.

7.3 Feedback from the Unit

The ontology has been presented to participants in, and closely related to, the unit at 7 occasions (Jun 06, Jun 18, Dec 13, Feb 20, Mar 14, April 11, April 22) in oral form (workshops, seminars, presentations, status meetings) and in written forms (posters, spreadsheets, slides), sometimes as categories sometimes in roadmap form.

The general feedback is that the approach is interesting and promising and that strategy work at this level of detail would be a valuable supplement: inquiries and qualitative data analysis is already used for improvement work, but not to this extent; particularly appreciated was that needs and features are anchored in data sources, and that the knowledge database is common, shared, and updateable.

Several work situations that could benefit from having an ontology were recognized by the unit: engineers and managers could quickly get ideas about what is on the market and what people would like to see in their processes; engineers and managers communicating with tool vendors would have a checklist of requirement areas and specific requirements to point at; process engineers could use the areas to subdivide and coordinate process improvement efforts; technology boards would have a good starting point for their analyses; engineers would find it easier to relate to technology strategy work. The approach was found promising not only for introduction of MDA, but software process improvement work in general.

Naturally, there was also critique: acute needs that engineers face every day (such as better layout-editors) tend to dominate over the long-term needs (such as better separation of concern in models); the sheer size of the roadmap made it difficult to comprehend; needs were too plentiful and detailed to guide improvement work; it was not obvious how an ongoing analysis activity at this level of technical detail would fit in the present organization; some unsatisfied identified were already satisfied in other units; the needs model was incomplete with respect to the project as a whole; needs were not ranked according to benefits to the organisations (some would lead to really big savings); to be really useful, a roadmap should also contain strategic information telling what to improve next.

8 Threats to Validity

Our research design is sensitive to following sources of errors, many of which are intrinsic to interpretative, case study research: (e_1) both needs and features are dynamic entities, i.e., the ontology is an interpretation of a snapshot of a situation that will change; (e_2) needs-related sampling is restricted to 7 out of 100 engineers, so the characterization of needs is hardly complete; (e_3) feature-related sampling is restricted to survey publications, thus recent advancements may not be represented; (e_4) the needs are influenced by the conceptions of the informants, and (e_5) the analysts; (e_6) the features are influenced by the conceptions of the analysts, and (e_7) the authors of the survey publications; (e_8) sampling and interpretation may be consciously or unconsciously biased to researcher concerns; (e_9) analytic generalization from a single project within a single unit may yield an ontology incomplete with respect to improvement areas of other projects, and (e_{10}) yield areas associated with the unit studied rather than complex embedded systems projects in general.

The following precautions have been taken to reduce the effect of these sources: for e_1, abstraction has been used (in the formulation of needs, features, and areas) to eliminate conceptual details not likely to withstand time; for e_2 we have chosen

informants with long experience in modelling, a wide perspective of the problems, and complementary views; for e_3, we only chose (at the time) recent survey publications; for e_4, we used open questioning from many perspectives and interpretation in search for indirectly expressed needs; for $e_{2\text{-}3}$ we developed a method to detect gaps in roadmaps; for $e_{5\text{-}6}$, needs, features, areas, and their relationships have been incrementally modelled using a qualitative data analysis tools and a common case study database shared by all analysts; analysts have met in four half-day sessions to discuss and revise concepts; concepts have been put to test in three analysis tasks; for $e_{7\text{-}8}$ emerging concepts have been presented to the unit, and the feedback taken into account. To handle $e_{9\text{-}10}$ requires a multiple case study, which is future work.

As for the validity of our approach: we have handled the critique of the roadmap by positioning the use of our ontology in software process improvement work as a well-developed starting points for recurring, organization specific ontology developpment in a changing world, rather than a universal ontology carved in stone; to investigate whether recurring ontology development is a cost effective complement to present process improvement activities is future work.

9 Related Work

The use of knowledge modeling techniques in software process improvement work is standard (see [23] for an overview), as is the use of ontologies for knowledge modeling [24]. We are not aware of any ontologies that give an empirically grounded holistic characterization of the improvement areas associated with introduction of MDA: Lange identifies 8 areas of activities supported by UML modeling tools [20] (which have been included in our ontology), but does not cover model content and managerial aspects; Störrle identifies 6 general areas and 18 specific operations of models [25], but does not characterize managerial and content aspects.

10 Summary and Conclusions

We have identified needs and features of concern to requirement engineers, system engineers, architects, designers, developers, and test engineers, proceeding from informal UML-based software development to MDA in the context of complex embedded systems development within Ericsson (Table 1–2). We have categorized the needs and features found to obtain a simple, informal ontology that defines improvement areas and subareas concerned with model content, modeling activities, and the management of modeling projects (Table 3-4). We have exemplified how the ontology may be used for roadmapping (Sec. 7.1), and defined methods to detect gaps in knowledge about features and needs (Sec. 7.2). We have identified 4 specific work situations within Ericsson that would benefit from using the ontology (Sec. 7.3). We have assessed the validity of the ontology for practical use (Sec 8.) and found it to be relevant and potentially useful. This, and experiences of ontology- driven and roadmap driven improvement work in other fields, allow us to conclude that, with the provided ontology, analysis methods, and underlying roadmap strategy at hand, MDA introduction in complex embedded systems projects becomes easier than in the absence of these aids.

Future work includes refining categories to reduce conceptual overlap, testing the ontology in actual software process improvement work, incorporating needs and features relevant to other units and organizations, and studying the cost effectiveness of the approach.

Acknowledgments. This work has been supported by Ericsson Software Research through Ericsson's Software Architecture and Quality Centre (SAQC).

References

1. Churchman, C.W.: Wicked problems. Management Science 14(4), 141–142 (1967)
2. Stahl, T., Völter, M.: Model-driven software development: technology, engineerig,management. Wiley, Chichester (2006)
3. Yin, R.K.: Case study research: design and methods. Sage Publications, ThousandOaks (2003)
4. Walsham, G.: Interpretive case studies in IS research: nature and method. Eur. J.Inf. Systs. 4(2), 74–81 (1995)
5. Wohlin, C., Höst, M., Henningsson, K.: Empirical research methods in softwareengineering. In: Empirical methods and studies in software engineering. LNCS,vol. 2765. Springer, Heidelberg (2003)
6. Strauss, A., Corbin, J.: Basics of qualitative research: techniques and procedures for developing grounded theory. Sage Publications, Thousand Oaks (1998)
7. Phaal, R., et al.: Technology roadmapping –a planning framework for evolution and revolution. Technological forecasting and social change 71, 5–26 (2003)
8. van Rees, R.: Clarity in the usage of the terms ontology, taxonomy and classification. In: CIB workgroup 78 conference, Auckland, Australia (2003)
9. Adler, N., Shani, A.B., Styhre, A.: Collaborative research in organizations. Sage Publications, Thousand Oaks (2004)
10. Jacobson, I.: Object oriented software engineering: a use case driven approach. Addison-Wesley, Reading (1992)
11. Rockstrom, A.S.: SDL-CCITT specification and description language. IEEE Transactionson communications 30(6), 1310–1318 (1982)
12. Jacobson, I.: Object-oriented development in an industrial environment. ACM SIGPLAN Notices 22(12), 183–191 (1987)
13. Börjesson, A.: Making software process improvement happen, Doctoral dissertationIT University of Gothenburg (2006)
14. Baker, P., Loh, S., Weil, F.: Model-driven engineering in a large industrial context -a Motorola case study. In: Briand, L.C., Williams, C. (eds.) MoDELS 2005. LNCS,vol. 3713, pp. 476–491. Springer, Heidelberg (2005)
15. France, R., Rumpe, B.: Model-driven development of complex software: a research roadmap. In: 29th Int. conf. on software engineering, Minneapolis, USA (2007)
16. FP6-IP 511731 MODELWARE D1.1-2 QoS Support in MODELWARE (2006)
17. FP6-IP 511731 MODELWARE D2.2 MDD Engineering Metrics Definition (2006)
18. FP6-IP 511731 MODELWARE D2.3 MDD Maturity Levels Definition (2006)
19. FP6-IP 511731 MODELWARE D2.5 MDD Engineering Metrics Baseline (2006)
20. FP6-IP 511731 MODELWARE D2.6 MDD Maturity Model (2006)
21. FP6-IP 511731 MODELWARE D2.8 MDD Process Framework (2006)
22. Miles, M.B., Huberman, A.M.: Qualitative data analysis. SAGE Publications, Thousand Oaks (1994)
23. Komi-Sirviö: Development and evaluation of software process improvement methods. Doctoral dissertation University of Ooulu (2004)
24. Djurić, D., Gašević, D., Devedžić, V.: Ontology modeling and MDA. Journal of bject Technology 4(1), 109–128 (2005)
25. Störrle, H.: A PROLOG-based Approach to Representing and Querying UML Models. n: Workshop on Visual Languages and Logic (VLL), Coeur d'Al`ene, Idaho, USA (2007)-