An integration centric approach for the coordination of distributed software development projects

Lars Taxén *

Department of Science and Technology, Campus Norrköping, Linköping University, Norrköping, Sweden

Received 27 October 2005; accepted 11 January 2006
Available online 9 March 2006

Abstract

This paper presents an approach for Distributed Software Development (DSD) that is based on two foundations. The first one is an integration centric engineering process, which aims at managing crucial dependencies in DSD projects. The second foundation is a strategy for operationalizing the coordination of the engineering process. The purpose of this strategy is to simultaneously provide global information system support for coordination and achieve common understanding about what should be coordinated and how. The approach has been successfully used at Ericsson, a major supplier of telecommunication systems worldwide, for coordinating extraordinary complex projects developing nodes in the third generation of mobile systems. Although many obstacles have to be addressed, the results indicate that the approach is a viable way to manage DSD during very demanding circumstances.

Keywords: Distributed software development; Integration centric engineering; Coordination; Common understanding; Flexible IS/IT support; Telecom systems

1. Introduction

The interest in distributed software development (DSD) has increased due to factors such as reduced costs, the access to well-educated labour pools, the possibility of 24 h development, global presence and proximity to customers, etc. e.g. [4,6,7,10–13, 15,16,23,25–27]. However, many challenges already present in centralized software (SW) development are aggravated by the distribution. At the core of these challenges lies the issue of coordination. “While there is no single cause of the software crisis, a major contributor is the problem of coordinating activities while developing large software systems.” ([16], p. 69).

The purpose of this paper is to discuss how Ericsson, a major supplier of telecommunication equipments worldwide, has approached DSD. The development of a telecom system can be seen as a paradigmatic example of many challenges involved in DSD. First, the telecom network is a truly complex system, which has been called the world’s largest machine. It consists of interacting nodes, each of which performs some kind of utility like keeping track of the position of a cellular phone, providing charging functions, etc. A variform of technologies such as SW, hardware, mechanical, optical, etc. are used. Legacy systems are mixed with new ones at an accelerating pace of change.

Second, a project developing a node in the third generation of mobile systems network can take more than a year to execute and involve several thousand participants. The number of software code lines may be in the order of millions. The development is in general carried out at many different sites worldwide, which usually have some autonomy to structure their own way of working.

As a result, the coordination of telecom development projects must deal with a multitude of technical, market related, cultural and organizational interdependencies. Most often, this requires mutual adjustment across many types of both technical and organizational boundaries [1].

The approach Ericsson has elaborated to DSD is based on two foundations. The first one is an integration centric engineering process, which aims at managing critical dependencies in the projects. The second foundation is a strategy for operationalizing the coordination of the engineering process. The purpose of this strategy is to simultaneously

* Address: Rundan 91, 14645 Tullinge, Sweden. Tel.: +46 730977864. E-mail address: lars.taxen@telia.com.

1 This type of development has also been named Global Software Development [6] and Offshore development [27].

0950-5849/$ - see front matter © 2006 Elsevier B.V. All rights reserved. doi:10.1016/j.infsof.2006.01.007

2 By operationalization, we mean the transformation of general definitions or theories into elements that can be manipulated, measured or observed in a particular situation in order to influence this situation.
provide global information system (IS) support for coordination and achieve common understanding about what should be coordinated and how. In this paper, we will refer to the ensemble of these two foundations as the Integration Centric Development (ICD) approach.

The paper is organized as follows: in Section 2, we give an account of the research design. In Section 3.1, we describe the engineering process. The core of this process is a construct called the ‘anatomy’, which is an illustration—preferably on one page—of the main functional dependencies in the system. The anatomy is the basis for building the total functionality of the SW system in steps—increments—that are consecutively verified and integrated.

In Section 3.2, we outline the strategy for operationalizing coordination. This strategy, which is called the Domain Construction Strategy (DCS), involves various management activates such as project management, requirement management, engineering change order management, baseline and milestone management, test management, system build management, etc.

In the following section, we report on some results from applying ICD in the Ericsson development practice. We discuss how ICD addresses important issues in DSD as well as problems with and limitations of ICD. Although many issues still have to be addressed, the main conclusion is that the ICD approach is a viable way to coordinate DSD during very demanding circumstances.

2. Research background and design

The background of the research is my professional work as a method and tools coordinator during more than 30 years at Ericsson. In the early 1990s, I began to systemize observations from the Ericsson practice regarding the coordination of development projects. These observations were eventually organized into a coherent ‘framework’ in which IS support, the DCS and the anatomy-based engineering process are important elements. During my PhD studies, I elaborated the ‘Activity Domain Theory’ in order to ground the framework theoretically [33,34].

Consequently, the research design can be seen from a variform of perspectives. Seen as a longitudinal case study, the research concerns empirical observations from the Ericsson practice between the early 1990s until the end of 2002. Another research perspective is theory development, in which the Activity Domain Theory and empirical findings were mutually influencing each other between approximately 1998 and 2003. Since I was actively participating in promoting the framework during this period, the research can also be classified as action research [3].

Finally, the research can be seen as a single-case, multiple units of analysis case study of the effects of the framework intervention in the 3G development practice [39]. The study, which is the focus of this paper, took place between roughly 1997 and 2003. It is classified as single-case since it concerns one organization and one case only. Given that the effects are studied both from the framework as a whole as well as and its constituting elements, it is classified as a multiple-unit study.

The nature of the research questions implies that the knowledge contribution of the study is of the explanatory type. Thus, no specific hypotheses were formulated in advance. The data sources are 14 interviews with participant in various roles such as project managers, method and tools coordinators, configuration mangers, etc. The interviews, which lasted between 1 and 2 h, were transcribed and checked with the interviewees for errors and misunderstandings. In addition, about 50 Ericsson internal documents, meeting notes, etc. were used. Finally, my own observations and notations over more than 10 years were included in the empirical data. The data analysis was carried out using a modified version of Grounded Theory [32]. The open coding resulted in over 100 categories grounded in more than 400 excerpts from interviews and documents.

In this paper, it is only possible to sketch the research design. A thorough account is given in [33], where issues like validity, reliability, relevance, bias, etc. are discussed.

3. The integration centric development approach

In this section, we will describe the two foundations of the ICD approach: the anatomy-based engineering process and the Domain Construction Strategy.

3.1. The anatomy-based engineering process

The basic motivation for the anatomy is the need in complex situations to understand how things depend on each other. Experiences from many projects at Ericsson indicated that traditional engineering processes and project management techniques did not address this problem well enough.

The anatomy is an illustration—preferably on one page—that shows the functional dependencies in the system from start-up to an operational system [1,2,18,36]. Here, the term ‘functional’ shall be understood as the capability of a certain system element to provide a utility that other system elements need. For example, the operating system of a computer needs a power source and a bootstrapping procedure in order to work. The gist of the anatomy-based process is to design and test the system in the same order as the functions are invoked. In a metaphorical sense, this can be seen as the order in which the system ‘comes alive’, hence the term ‘anatomy’.

The particular perspective on dependencies provided by the anatomy emerged over many years [1]. Thus, the specific form and usage of the anatomy was chiselled out in the development practice. The theoretical underpinning of the anatomy construct is less developed. In [36], Taxén and Lillieskold argue that the anatomy and its associated plans provide different stakeholder in the project with a common understanding about which actions they can take and what the consequences of these actions are. Thus, the engineering process can be conceived of as a way for various stakeholders to reconcile their actions.
The engineering process is executed in three steps: *anatomy definition*, *increment planning* and *integration planning*. In the following sections, we will describe these steps using a simplified example from the Ericsson practice: the development of a processor in a telecom system.

### 3.1.1. Anatomy definition

The purpose of the first step is to achieve a common understanding about how the system works in terms of functional dependencies. In Fig. 1, the anatomy of the example processor is shown. The boxes indicate functions and the lines dependencies, which should be read from the bottom of the figure to the top.

The prerequisite for the anatomy definition is a requirement specification of what the system should provide. Based on this, the anatomy is created in several meetings where the mindset should be: ‘if you “power-on” what happens then and then’. This question is repeated until the end functionality is reached. When deciding which functions to include, the focus should be on integration and testability since this will be used to plan and monitor the project.

The main stakeholders in the anatomy definition phase are system architects. In addition, other stakeholders such as customers, project managers, requirement managers, etc. will be included whenever pertinent. Thus, the anatomy can be apprehended as the manifestation of a common understanding about the architecture of the system. However, it is important to realize that the definition of the anatomy is dependent on both the system and the participants. Other participants might have included different functionalities and chosen a different level of detail.

### 3.1.2. Increment planning

The purpose of the second step is to outline the implementation of the system. The functions are grouped into development and integration steps—increments—in such a way that the additional functionality after each increment is executable and verifiable. The intention is to parallelize design and testing as much as possible. The distribution of the development work is determined by a number of circumstances such as available resources, customer feedback, complicated or simple functions, geographical proximity between resources, functions that can be tested jointly, etc. [1]. In Fig. 2, a possible increment plan of the example processor is shown.

The main stakeholders in increment planning are system integrators, system testers and project managers. In this step, the functional dependencies between increments are brought to the fore. Functions and dependencies within each increment are subdued.

### 3.1.3. Integration planning

In the third step, the purpose is to divide the work between subprojects and establish a common understanding about what is delivered from whom and when. The main stakeholders are project managers. Resources are assigned and dates for deliveries of the increments to system integration are negotiated. For each increment, traditional time and resource plans are made. The dependencies in focus are those between subprojects.

During the project, the plan is used as an instrument for communicating the progress of the project. The state of each increment is visualized by traffic-light cues such as Green—On Plan, Yellow—Warning, Red—Off Track, etc. Impacts of delays are clearly shown, which give the project management time to take corrective actions. In Fig. 3, an integration plan of the example processor is shown.

At a quick glance, the integration plan appears similar to traditional PERT diagrams in the sense that it shows dependencies between tasks. However, the way the integration plan is constructed is very different from the construction...
of PERT diagrams. Moreover, the focus of the integration plan is on expressiveness and ease of use, the ultimate purpose of which is to manage complexity. A comparison of various project-planning techniques from this point of view is given in [8].

In summary, the anatomy-based engineering process uses a simple illustration—the anatomy—as a mechanism to achieve common understanding among different stakeholders about how the project should be coordinated. This is of utmost importance when frequent re-planning of the project is needed. In doing so, all the steps in the process may be affected. Thus, these steps should not be seen as strictly consecutive. Rather, they should be regarded as various interrelated perspectives, which make it possible for stakeholders to manage a complex situation.

3.2. The domain construction strategy

The purpose of the DCS is to operationalize the coordination of the engineering process. Coordination is a nebulous concept that has been defined in various ways, e.g., [17,19,24]. In ICD, the point of departure is the definition suggested by Malone and Crowston: ‘Coordination is managing dependencies between activities’ ([19], p. 90).

In order to operationalize this definition, Taxén has suggested to ground coordination in the *workpractice* construct.
In a workpractice, actors produce a certain result that other actors need [9,28]. By actors, we mean people working together in socially organized settings, for example, in the development of telecom systems at Ericsson. Some actions performed in workpractices have the explicit goal of coordinating the actions of other actors. By regarding the workpractice from a coordination point of view, we emphasize a particular perspective of the workpractice. We will refer to this perspective as the coordination domain.

The coordination domain has to become a social reality for the actors in order to operationalize coordination [30]. This reality has to be actively constructed in terms of common concepts, rules, norms, etc., as well as artefacts such as tools, documents, models, etc., which are meaningful in the domain.

In DCS, the construction comprises mainly two elements: a context model and an IS. The context model shows what items the actors consider relevant for coordination and how these items are related to each other. For example, in order to trace requirements to system elements fulfilling these requirements, the context model must include a requirement type, a system element type and a directed relation type connecting these types. In ICD, the context model must also include a function type and a relation signifying the dependencies between functions in order to enable the coordination of the anatomy-based engineering process.

By continually iterating between modifying the context model, implementing it in the IS and evaluating the result, coordination is gradually operationalized in terms of the context model, the IS support and a common understanding among the actors about what constitutes coordination. A detailed account for this approach is given in [33,35].

3.2.1. Elaboration

In this phase, the main purpose is to achieve a tentative common understanding about the structure and content of the coordination domain. The context model is defined in terms of types, type hierarchies, relationships, attributes, cardinalities, revision stepping rules, state sets, etc. In Fig. 4, an example from Ericsson of a context model from 1997 is shown. It can be seen that it includes requirement types (Set of Requirements) as well as anatomy related types (‘Functional Anatomy’, ‘Feature Increment’, etc.).

Next, the context model is implement the IS and objects of the various types in the context model are instantiated in the IS. When this is done, the information in the IS can be viewed, browsed, printed out in reports, etc. In Fig. 5, an example of a view corresponding to the context model in Fig. 4 is shown.

The information in the IS is evaluated with respect to its usefulness in providing coordination support. If this is not satisfactory, the context model is modified and implemented anew. This is continued until the participants agree that the constructed domain is meaningful and useful.

3.2.2. Trust boosting

The purpose of this phase is to boost the trust about the feasibility of the domain as constructed in the elaboration phase. Key issues are getting all participants in the project to trust the data in the IS and make sure that the performance of the IS is acceptable at all units worldwide. This is done in one sharp project, that is, a project that develops a product for some client. An important task in this phase is to make sure that
the participants use the IS. This can be achieved by requiring that the only source for progress reports is the data in the IS.

3.2.3. Expansion

In this phase, several projects are involved. The purpose is to establish the coordination domain as an institutionalised way of working, which does not fade away when the projects are finished. Modifications in the context model and the IS implementation are done by controlled changes. If the domain has to cooperate with other domains, a work must be initialised to coordinate these domains.

4. Results

The study related in this paper concerns the effects of the ICD approach in the development of the third generation of mobile systems between approximately 1999 and 2003 [33]. Around 140 main projects and subprojects were involved. Two main coordination domains were constructed during this period: the S-domain in Stockholm, Sweden and the A-domain in Aachen, Germany. These domains developed different parts of the 3G system. The coordination of the projects was supported by domain specific IS applications built on the same platform, the Matrix PDM system from Matrix-One [22].

In order to illustrate the complexity of the development, the integration plan of one node in the 3G system, the so-called Mobile Switching Centre, is shown in Fig. 6. This node was developed by 27 subprojects distributed over 22 development units in altogether 18 countries.

In order to illustrate the complexity of the development, the integration plan of one node in the 3G system, the so-called Mobile Switching Centre, is shown in Fig. 6. This node was developed by 27 subprojects distributed over 22 development units in altogether 18 countries.

In order to illustrate the complexity of the development, the integration plan of one node in the 3G system, the so-called Mobile Switching Centre, is shown in Fig. 6. This node was developed by 27 subprojects distributed over 22 development units in altogether 18 countries.

The study related in this paper concerns the effects of the ICD approach in the development of the third generation of mobile systems between approximately 1999 and 2003 [33]. Around 140 main projects and subprojects were involved. Two main coordination domains were constructed during this period: the S-domain in Stockholm, Sweden and the A-domain in Aachen, Germany. These domains developed different parts of the 3G system. The coordination of the projects was supported by domain specific IS applications built on the same platform, the Matrix PDM system from Matrix-One [22]. The main reason for developing two separate IS applications was different opinions about how to construct the domains. These opinions were not reconciled during the study period.

In order to illustrate the complexity of the development, the integration plan of one node in the 3G system, the so-called Mobile Switching Centre, is shown in Fig. 6. This node was developed by 27 subprojects distributed over 22 development units in altogether 18 countries.

The white squares signify increments and the lines between them dependencies. In addition, dates for integration, responsible projects, status of the increments, major deliveries, etc. are included. In contrast to the example in Fig. 3, the dependency order in Fig. 6 is from the top towards the bottom of the figure.

The elaboration phase of the S-domain occurred approximately between 1998 and 1999. The work was carried out by this author, another Ericsson employee and two IS consultants. In addition, user representatives such as project managers, requirement coordinators, configuration managers took part in the development. The result was unrivalled in the history of Ericsson. In 19 months, the following coordination support was implemented:

- A common project repository for sites distributed to Sweden, Italy, Australia, Norway and Croatia.
- Support for requirement management.
- Support for engineering change order management.
- Support for baseline and milestone management.
- Support for test configuration management.
- Support for the anatomy-based engineering process.
- Support for product and document management for the needs of the project.
- Three general report generators.
- An interface to the Ericsson common document library.
- An application specific web-client in order to improve performance and stability.
• Support for complete traceability between coordination items.
• On line monitoring of the status in the project.
• Basic Matrix functionality such as logging all events in the IS, defining user roles and access rights, security checks, etc.

The A-domain was elaborated between the end of 1999 and mid-2000 with similar results. The effects on coordination were profound [33]. For example, some project managers claimed that the development of the node in Fig. 6 would not have been possible without the ICD approach:

Especially for the execution part I think we would not have been able to run this project without the tool. I think if you simply look at the number of work packages [increments], the number of products that we have delivered, the number of deliveries that we have had, if we would have had to maintain that manually, that would have been a sheer disaster. (Project manager 4, A-domain).

In spite of the positive results, the ICD approach became a one-time shot at Ericsson. The anatomy-based engineering process emerged at Ericsson in the early 1990s [1]. However, before 1999 the IS support based on Matrix and the Domain Construction Strategy did not exist. After 2002, the coherent ICD approach began to disintegrate. At that time, the IT crisis hit Ericsson hard. Unification, centralization and concentration strategies were enforced by top management as perceived ways to reduce costs. Towards the end of this study (2003), work was in place to consolidate the separate domains into one central domain for the entire Ericsson organization. The context models were merged into a ‘Common Ericsson Model’ maintained by a line organization and, consequently, detached from the needs of the development practices ‘in the field’. The close interaction between users and developer inherent in ICD was replaced with a more formalized procedure, where IS applications were ordered by one group and implemented by another group. The final step in the cost reduction strategy was the outsourcing of the IS application development to an organization outside Ericsson.

In essence, most of the elements in the ICD approach remain operational at Ericsson: the anatomy-based engineering process, the Matrix IS support, the context model, etc. However, the idea of several coordination domains within the same organization and the Domain Construction Strategy are dismantled. The prevailing way of thinking is amply expressed by the banner ‘one company–one process’. Thus, the coherent ICD approach as described in this paper does no longer exist at Ericsson.

5. Discussion

In complex development situations, there is a need to concentrate on a limited number of problems areas. The stance taken in ICD is that the following challenges need to be confronted in DSD:

- Support for complete traceability between coordination items.
- On line monitoring of the status in the project.
- Basic Matrix functionality such as logging all events in the IS, defining user roles and access rights, security checks, etc.

Towards the end of this study (2003), work was in place to consolidate the separate domains into one central domain for the entire Ericsson organization. The context models were merged into a ‘Common Ericsson Model’ maintained by a line organization and, consequently, detached from the needs of the development practices ‘in the field’. The close interaction between users and developer inherent in ICD was replaced with a more formalized procedure, where IS applications were ordered by one group and implemented by another group. The final step in the cost reduction strategy was the outsourcing of the IS application development to an organization outside Ericsson.

In essence, most of the elements in the ICD approach remain operational at Ericsson: the anatomy-based engineering process, the Matrix IS support, the context model, etc. However, the idea of several coordination domains within the same organization and the Domain Construction Strategy are dismantled. The prevailing way of thinking is amply expressed by the banner ‘one company–one process’. Thus, the coherent ICD approach as described in this paper does no longer exist at Ericsson.

Fig. 6. The integration plan of the Mobile Switching Centre node.
• Achieving common understanding.
• Clarifying dependencies.
• Distributing tasks.
• Engineering requirements.
• Planning and controlling the project.
• Managing cultural diversity.
• Providing proper IS/IT support.
• Managing change.

In the following, we shall discuss how ICD as a whole addresses these areas. We will support our claims with stakeholder statements from the 3G development projects. In addition, we will discuss the relation of SW development models to ICD as well as problems and limitations of the approach. In Fig. 7, the framing of the discussion is illustrated.

5.1. Achieving common understanding

By common understanding, we refer to a common stock of knowledge, which is meaningful for the actors and enable them to coordinate their work. Achieving common understanding is an arduous task [20]. The difficulties emanate from the fact that meaning is constructed in social activity [30,38]. The criteria by which meanings are valued is ‘usefulness’ rather than ‘correctness’ or ‘truth’. As long as a meaning of a phenomenon such as a word, sign, artefact, etc. brings about actions with intended consequences, the phenomenon is apprehended as meaningful in that practice. When misunderstandings or disagreements occur, the meaning must be reconsidered and reflected upon.

The need for common understanding of key elements in DSD is emphasized in the literature, e.g. [4,12,15,23,25–27]. One such element is the architecture: ‘The coordination of multi-site work needs a common understanding of the architecture of the system to direct the development the work toward a coherent, working system’ ([25], p. 245). The architecture should be transparent in the sense that it can be easily communicated across the entire project [4,26].

The basic mechanism in ICD for achieving common understanding is the experiential way of working where reflection and action are continuously interwoven [14,21,29]. One manifestation of the common understanding is the anatomy, which signifies the architecture of the system. Other manifestations are the increment and integration plans. In order to alleviate the understanding, the anatomy and its associated plans are all illustrated on a single page, which can be placed on a wall in a project office or displayed on the web. In addition, common understanding from the DCS is manifested in the context model and its implementation in Matrix. These various manifestations provide a common view of the project for all participants, regardless of how they are distributed across geographical and organizational borders:

For we didn’t have one object type just being on its own, it was always to show impacts of relationships to other, and [...] all the different roles together, they build the overall picture then. (Method and tools coordinator 3, A-domain).

5.2. Clarifying dependencies

When coordinating complex system development tasks, it is crucial to know how things depend on each other [19,37]. Some of these dependencies are expressed by the anatomy, which enables the actors to take proper actions:

And also based on the anatomy chart [...] you see a lot of dependencies both time wise and product wise. [...] Of course that is also a key issue. (Project manager, A-domain).

In addition to the functional dependencies, the implementation of the context model in Matrix makes it possible to manage other types of dependencies:

I think for the MSC [Mobile Switching Centre node] there is a clear need for the tool and I have met now a couple of people who have said: without the tool we would not have
survived the projects. [This is due to] the complexity of the node, the complexity of the dependencies, time pressure people have. If you visualize dependencies it is far easier to take decisions. (Project manager 3, A-domain).

Another aspect of dependencies is traceability, for example, from requirements to system elements and test cases. Traceability is provided by browsing the information in Matrix or generating tailor cut reports with traceability information. The scope of traceability is defined in the context model, where the types and their relationships frame what can be traced.

5.3. Distribution of tasks

When distributing the development tasks, group cohesion and task coupling should be considered [27]. Group cohesion refers to variations in language, culture, organization, etc. A group with large variety in these respects is defined as having low group cohesion. Task coupling refers to the need of co-presence, proximity and face-to-face communication.

The concepts of group cohesion and task coupling may be utilized when defining the increments in ICD. The obvious choice is to define increments, which delineate tasks with high group cohesion and high task coupling. Moreover, subprojects in the integration plan may be allocated to groups having a high degree of cohesion. Furthermore, increments borders should be placed at nodes in the anatomy, where a large number of dependencies converge or diverge. Either such nodes indicate opportunities for parallel development (diverging dependencies) or potential critical tasks (converging dependencies) on which many other tasks depend.

The definition of the anatomy requires a lot of interaction, communication and face-to-face meetings among participants from various sites. Thus, anatomy definition has a high degree of task coupling and low group cohesion. By participating in this work, the participants will communicate the common understanding of the anatomy at their home sites. In other words, they will act as liaisons in the project, something that have been identified as quite important in DSD [4].

The anatomy and increment plan can be utilized interchangeably when distributing tasks. For example, the architecture of the system can be defined in such a way that it alleviates the definition of tasks with low coupling. This is of course only possible in situations where the architecture is not settled.

5.4. Requirements engineering

Requirements engineering has been identified as a main source of error in software development [6,15,26]. Several features of ICD address this issue. First, the anatomy-based engineering process was originally developed in order to mitigate the effect of requirement instability. Stable requirements can be allocated to early increments and volatile ones to later increments. Impacts of requirement changes may be isolated to a limited set of increments. Second, requirements traceability is provided by the IS. Finally, changes in requirements are subject to the engineering change order process, which also is supported by the IS. In short, ICD provides ample support for requirement engineering:

I think that we have the possibilities to manage requirements in a good way and make them obvious and we can achieve a very clear traceability all the way from customer requirements one might say. (Project manager 2, S-domain).

5.5. Project planning and control

DSD brings about more demanding requirements on project management [15]. When people cannot meet face to face, the burden of coordination and communication increases. The geographical separation means that effort estimations, project planning and control become more difficult. In addition, the management of issues like trust, responsibilities, commitments, incitements, etc. is aggravated in DSD.

5.5.1. Communication

The increment and integration plans are the main project communication mechanisms in ICD. Based on these plans, responsibilities, progress, delivery dates, release plans, etc., can be communicated in a uniform manner to the entire project.

Another communication mechanism is the various report generators in Matrix (see Fig. 8). The report shows different increments (‘Work Package’) and their dependencies to other increments (‘Dependent WPs’). The column ‘Connected Blocks’ indicates which software products implement the functionality in the increment. In addition, some other information regarding specification documents (‘IPs’), slogans, etc., is shown. These reports provide a common picture of the status of the project:

I presented this way of working to them, to the mobile people. And [they thought] this is nice and simple and easy to understand for everybody and very effective… the project managers [got] the information they needed directly from the web with one click in these pre-defined tables. (Method and tools coordinator 4, A-domain).

5.5.2. Coordination focus

Besides providing the overall coordination support, one important effect of ICD is that coordination efforts can be more focused:

“Yes, what is the great benefit is that you have one common place where all the project area stored the information. It means that a lot of the coordination, which previously went via the main project, now can go directly. A lot of coordination is now happening on the level it should be.” (Project manager 4, A-domain).

This meant that that the need for the main project management to engage in coordinating subprojects decreased.
5.5.3. Commitments and responsibilities

ICD enhances commitments and responsibilities. The increment and integration plans show which groups are responsible for what as well as the dependencies between these groups. These plans, together with their implementation in the Matrix tool, make it more difficult for the participants to escape their responsibilities:

I will never forget I was in a meeting where they were actually pointing in a circle. Nobody in this meeting had a problem. But at the end of the day, the build process didn’t work. And without a proof that it is their problem they would not do anything about it. Now this tool makes it so damn visible that they have a problem. (Project manager 3, A-domain).

5.5.4. Separation of concerns

The focus for total project management is directed towards the status of each increment and the dependencies between them. On the other hand, the focus of subproject managers and team leaders is on the inner workings of each increment. The ICD makes it possible to change the focus between these different levels:

Of course there is also something that is maybe also a benefit of the tool is that we do have different levels of projects. And the tool can really provide support for various levels, [...] It is one common database with everything in it… that is very valuable. (Project manager 4, A-domain).

5.5.5. Trust

A major difficulty in both coordination domains was to convince the participants of entering data into Matrix. There were various reasons for this: lack of support resources, resistance to change established ways of working, poor performance of early versions of Matrix, awkward user interface, etc. However, a main reason was that the participants did not understand why they had to enter the data. From their point of view, entering data into the IS was just an extra burden. This situation created a sense of distrust regarding the data in Matrix.

The mechanism to boost the trust is to create a chain of dependencies among the participants in such a way that if someone fails to enter the data, the others will suffer:

As soon as different roles take the reports as input for their activities you must trust them. That is the driver to motivate all the people to enter the information into the tool. (Method and tools coordinator 3, A-domain).

5.6. Managing cultural diversity

The issue of culture in relation to organizations has been extensively treated in the literature, e.g. [31]. In ICD, one aspect of cultural diversity is in focus: different understandings at various sites involved in DSD. In essence, this concerns the balance between central and local ways of working.

There are different opinions about the role of commonality in DSD. For example, Ebert and De Neve from Alcatel recommend that all sites working with the same type of products should use common processes, methods and terminology [7]. On the other hand, Battin et al. from Motorola regard not imposing a common process as a key strategy in DSD [4]. Even though it is desirable to use the same tools, methods and processes throughout the project, this is seldom, if ever, an option. This is especially so if the DSD project is set up from a more or less temporary configuration of otherwise independent organizations.

The basic mechanism in ICD to address cultural diversity is the workpractice construct. The workpractice delimits the area within which common understanding emerges. In order for workpractices to interact, there must be some commonality between them. However, this commonality is limited to what is necessary for coordinating their activities.

In order to illustrate the balancing issue, we will compare the context models of the A and S-domains. These models can
be apprehended as the common understanding about what items are relevant in each domain. In Fig. 9, a high-level view of the context models for the S-domain is shown. The corresponding context model for the A-domain is shown in Fig. 10.

Both models were constructed in development work-practices with the same purpose: to develop telecom systems using the anatomy-based engineering process. Over several years, the context models were chiselled out and valued with respect to their usefulness in providing coordination support for the development. It is beyond the scope of this paper to describe the models in detail. However, it can be observed that they are very differently constructed. Even the most central item, the increment, was differently conceived (encircled in the figures). Names, attributes, life cycle states, etc., were constructed differently.

The key insight from this observation is that the rapid development of the coordination support was achieved by not enforcing a common way of working in both domains. This was confirmed in the analysis of the empirical data, for example, as in the following statement:

I know there are some initiatives and try to come up with one object model for all projects within Ericsson. And I think that’s a little bit trying to search for the Holy Graal. […] I mean you can even see how different the ways of working are within our subprojects at this moment […]. To even keep one common way of working for that is really quite a challenge (Project manager 4, A-domain).

The drawback of the autonomous ways of working was that the coordination between the A and S-domains was less attended. This coordination had to be done without tool support. Thus, the balance between centrality and locality was too much biased towards the locality side.

5.7. Providing IS/IT support

DSD puts higher requirements on the IS/IT support. There is a need for a common project repository where requirements, engineering change orders, system builds, etc., is globally available [10]. The infrastructure at distant sites may be inadequate, which may cause problems with IS performance and stability. In addition, problems related to tool support, upgrading and maintenance are aggravated in DSD [15].
5.7.1. Common repository for coordination data

In ICD, the common project repository is provided by Matrix. Having all coordination data in the same IS is advantageous in several ways:

- Interfaces between ISs in the coordination domain are not needed.
- The coordination information is consistent.
- There is a homogeneous tool worldview of the entire coordination domain.
- Changes, which are common to all coordination areas, are easier to implement.
- The ability to react to changes is higher.

There will still be a need to interface Matrix to other tools, for example, special purpose tools for configuration management of SW code files during the development, or Enterprise Resource Planning (ERP) tools. However, interfaces to these tools will be on the border of the coordination domain rather than inside it.

5.7.2. Performance and stability

In each of the A and S-domains, a central Matrix server was accessed from clients at the development sites. However, this architecture was at times faced with severe problems. The performance of the early versions of Matrix turned out to be insufficient for global access. The reasons were poor quality of the IT-infrastructures and inadequate technical solutions in the Matrix system. Other problems were a low acceptance of the user interface of Matrix and poor training in using the tool, particularly at remote sites. These problems were solved in later versions of Matrix, which were based on low interaction, application specific web-clients. Thus, a major lesson is that issues like performance and stability should be secured early in the construction of the coordination domains.

5.8. Changes

In ICD, two types of changes can be identified. The first one concerns the elaboration phase in the Domain Construction Strategy. In this phase, changes in the context model and its implementation in Matrix are made quite frequently. Between 1999 and 2002, around 200 and 500 changes were made in the A and S-domains, respectively. A necessary prerequisite for this type of change is the ease by which the implementation in Matrix can be modified. However, frequent changes have the drawback that the implementation may turn out to be a patchwork, which has to undergo major reconstructions. This happened once in the A-domain. The experience was, nonetheless, that this problem could be mastered.

The other type of evolution concerns a long-term evolution, which is related to influences from external changes, adding new coordination support, providing maintenance, etc. The results indicate that this type of evolution is possible in ICD:

Well, the positive effect is that we have an integrated project support system where we have tremendous possibilities to improve, continuously improve our operations (Project manager 2, S-domain).

5.9. ICD and software development models

DSD implies that the software development model must be carefully chosen. Sakthivel [27] has shown that there is a complex relationship between the potential for distribution and the type of development process (spiral, waterfall, Rapid Application Development, Agile Development Method, etc.).

ICD does not presume a particular SW development model. In fact, the functionality in an increment may as well be implemented in hardware, which was the case for some increments in the 3G development. This means that the SW development can be based on different models. Thus, when distributing the tasks, site-specific experiences, development traditions, tools, etc., can be taken into consideration.

5.10. Concerns

Some concerns with the ICD approach are as follows.

5.10.1. The ICD approach is considered too costly by top management

As related in Section 4, maintaining several coordination domains was considered too costly. At a first glance, this seems obvious for several reasons. The cost of the IS support becomes higher since more licences have to be acquired. The maintenance costs of the IS applications increases. Users trained in one domain cannot easily be transferred to another domain. Different ways of working may cause problems with misunderstandings and erroneous deliveries, and so on.

These aspects are all relevant and need to be carefully considered. After the crisis years, Ericsson is in fact doing quite well. This would indicate that the strategy of maintaining one coordination domain only is successful. However, costs associated with more intangible aspects, such as achieving a common understanding, is seldom, if ever, included in standard corporate analyses. These costs may be substantial. An analysis, which considers such costs, may indicate that several coordination domains are optimal. In fact, the results indicate that constructing a single coordination domain from the outset would not have been possible at all. Altogether, the issue of one versus several coordination domains needs to be elaborated in future research.

5.10.2. A workpractice view on the organization is needed

ICD presumes that the organization is apprehended in terms of workpractices, which have certain autonomy to construct their own ways of working. This may be problematic, especially during cost reduction efforts where centralization, uniformity and commonality are honoured. Thus, an alternative mind setting among top management concerning what constitutes an organization is needed.
5.10.3. Tool replacements may cause conflicts

Matrix is a common repository for data that traditionally are kept in separate tools for requirement management, engineering change order management, etc. As stated above, a common repository for such data is preferable in several ways. Thus, it is natural to replace the many tools by a single one. However, actually carrying out such a replacement may be problematic due to efforts already invested in existing tools and resistance from users and other stakeholders.

5.10.4. Conflicts with established IS strategies

Matrix may not be compliant with corporate IS strategies. For example, the organization may have decided for an ERP system such as SAP to support multi-site, common processes. Quite naturally, it is desirable to use this system in as many areas as possible. Even if SAP is unsuitable for the ICD approach due inadequate flexibility, it may be difficult to get acceptance for introducing yet another IS platform in the organization.

5.10.5. ICD challenges the traditional view on sound engineering practice

From a rational-based, traditional engineering point of view, many changes indicate a badly specified system. For example, the waterfall engineering process assumes that the requirements are defined and fixed before the analysis phase is commenced. In contrast, the very enabler of the fast and efficient construction of the coordination domain in ICD is not specifying the requirements in detail. The Domain Construction Strategy enforces an iterative IS development method where the implementation is frequently changed. Thus, the ICD approach requires a new mind setting concerning what characterizes good engineering practice.

5.10.6. Determining an optimal set of coordination domains

Identifying an optimal set of coordination domains may be problematic. On the one hand, the construction of common understanding is aggravated if the domain becomes too large and heterogeneous. On the other hand, many small domains bring about several IS implementations, which is costly to set up and maintain.

5.10.7. Coordinating coordination domains

The coordination between the A and S-domains was not implemented according to the intentions in the ICD approach. During the construction of these domains, there was no organizational unit in place at Ericsson to enforce the necessary commonality. Thus, a lesson learned is that the construction of new domains must go together with the construction of the coordination between them. This construction, which can be seen as a meta-coordination, can in turn be done according to the Domain Construction Strategy.

5.11. Limitations

To the best of our knowledge, the ICD approach has been in use only at Ericsson. This raises questions about the transferability of the results to other DSD enterprises. However, the development of telecom systems comprises most, if not all, of the challenges in DSD, which would indicate that the approach is indeed transferable. This is a topic for further research.

Another issue concerns the IS used in the Domain Construction Strategy. Again, the experiences so far come from one particular IS, the Matrix system. It is desirable to identify general criteria by which other IS could be evaluated for use in the ICD approach.

The coordination of software code files is a major issue in DSD [4,7,12,15]. Different groups may be concurrently working on variants of the same piece of code, which must be consolidated and synchronized for global integration. This aspect of DSD is outside the scope of ICD. However, there are several interaction points between the coordination domain and code writing activities. For example, when the code implementing the functionality in an increment is tested and verified, this should be notified to the coordination domain. This implies that Matrix and code configuration tools need to be interfaced [5]. However, it is outside the scope of this paper to discuss this matter.

6. Conclusions

The Integration Centric Development approach addresses many of the challenges inherent in distributed software development. Extraordinary complex nodes in the third generation of mobile systems have been successfully developed during very demanding circumstances. The experiences so far are limited to one company, Ericsson, where the approach was extensively used between 1998 and 2003. A main conclusion is that the approach is a viable way to coordinate distributed software development tasks where social issues such as communication, coordination and common understanding are of prime concern.

References
