2016

The learning of sociotechnical practices in embedded systems development

Christian Koch
Chalmers University of Technology, Sweden, Christian.koch@chalmers.se

John Bang Mathiasen
Aarhus University, Denmark, johnbm@btech.au.dk

The learning of sociotechnical practices in embedded systems development

Christian Koch,[1] John Bang Mathiasen[2]

[1] Chalmers University of Technology, Sven Hultinsgatan 8,
42196 Gothenburg, Sweden

[2] Aarhus University, Department of Business Development and Technology, School of Business and Social Sciences,
7400 Herning, Denmark

Abstract. Embedded systems are an important element of controlling wind turbines. Embedded systems development and learning are inseparable. This paper presents a study of wind turbine control systems and the developers' learning with the hope of contributing to the understanding of embedded systems development and practice-based learning. Our approach focuses on the sociotechnical practices, including the reciprocity between constitutive means and the developers' individual experience; central matters in this framework involve the process of understanding the indeterminacy and the handling of the IT. The embedded systems development case addresses development of hardware, software and their interfaces. Learning occurs, enabled by converging understanding of the indeterminate situation and the different experience of the developers. The case shows that embedded systems developers deliberately keep technical knowledge close to their chests, which constrains learning. The paper contributes to the understanding of the individual systems developer's learning and the mechanisms of enablers and obstacles in the learning processes.

Keywords: Embedded systems development, learning, Dewey, wind turbine control

1 Introduction

Embedded systems development is important for the realisation of contemporary ubiquitous solutions, including the internet of things, and considerable growth is expected in the years to come (9% according to Technavio [1]). Embedded systems development integrates hardware, software and physical design.

One central element of embedded systems development is successful learning of the developers themselves. Previous studies of learning in embedded systems development [2], [3] have often understood systems development and learning as independent of the particular type of system and as being straightforward [3], [4]. Practice-based learning studies [5], [6], [7] have reminded us that learning is not straightforward and have highlighted that development (here of embedded systems) context (here sociotechnical practice) and learning processes are inseparable, but in general these studies disregard individual learning. Hence, the purpose of this paper is to study embedded systems development and practice-based learning within interorganisational, cross-functional and daily sociotechnical practices, drawing on previous studies of individual learning [2], but focusing on the very micro trajectory of learning [8], [9], and asking the question:

What learning occurs in an embedded systems development project? And what enablers and constraints are there for learning in the development process?

The empirical work comprises an ethnographic study of embedded systems development. It involves the specifying and system-producing company (pen-name Hansen), a customer company (pen-name GlobeCorp), and two consulting companies (pen-names M-Consult and G-Consult). Embedded systems for wind turbine control combine hardware (HW) and software (SW) sub-systems, which are integrated into a larger mechanical, electrical system and equipped with appropriate input/output (I/O) devices [10]. The embedded systems have to cope with such complexities as the intermittent and seasonal variability of the wind powering the wind turbine [11]. The learning discussed here occurs in a commercial context, and the systems development is closely intertwined with business processes [10], [12]. The challenge is to create low-cost yet robust and quality embedded systems, where the embedded systems developers collaborate with sales, prototype production and management on a daily basis. The systems development activities include developing specifications, programming, testing, and prototyping as well as manufacturing the embedded systems; this paper focuses on developing HW, SW and I/O (interfaces).

The framework of understanding combines Dewey's [9] action-oriented learning approach with science technology and society studies [13], [14], [15], [16], [17], [18,] [19]. The Deweyan understanding of learning highlights the individual's experience and inquiring as central to the learning process and thus gaining new experience. Accordingly, this research adapts the practice-based learning approach applied by Brown & Duguid [5], Gherardi & Nicolini [6] and Nonaka et al [7], emphasising a situational practice for the development and learning processes, with the view that the individual developers have different levels of experience and commitment when carrying out the embedded systems development. This dual heterogeneous understanding is combined with the understanding of technology as text [13], [16], [18], as well as the view of working practices in which systems development occurs is a sociotechnical practice [14]; this notion is adopted to underline the mutual shaping of the social and the technical in embedded system development [19], [20]. "Doings" is a term conceptualised by Goffman [21] to understand the process that commences when an individual is in an indeterminate situation and tries to make sense of it. Since experience is embodied and yet inseparable from the sociotechnical practice, the doings are facilitated by reciprocal interaction between the developer's experience and accessible constitutive means within the perceivable sphere of the sociotechnical practice [9]. The gradual transformation of the indeterminate situation into a determinate situation requires a sequence of doings, defined as a strip of doings [21, p. 10]. Learning is therefore understood as moving through a strip of doings from an indeterminate to a determinate situation within a sociotechnical practice (from now on "practice").

This paper's contribution is the use of Dewey's approach to learning with a focus on micro-processes of the developer's learning in embedded systems development, thereby adding to the understanding of both individual and collective learning. By underpinning this approach with extensive longitudinal empirical fieldwork, the paper illustrates practice-based learning processes and shows that the often advocated close relationship with customers is overruled by the supplier's strategy of keeping customers at arm's length. Thus, we hope to contribute to the relatively few studies of embedded systems development with a practice-based view.

The paper is structured as follows: The framework of understanding for embedded systems development and learning processes is presented and discussed first, followed by methodological considerations. Then, the case of the embedded systems development is presented and analysed, followed by a cross-case discussion, implications and finally the conclusion.

2 Framework of Understanding

The development of embedded systems involves a complex interplay among SW, HW and other developers as well as several forms of constitutive means such as sketches, drawings and electrical diagrams. This interplay forms various compositions of the practices. The central interest of this article is to better understand how learning occurs when systems developers conduct embedded systems development activities. The presentation of the framework opens by discussing characteristics of embedded systems development. Then, embedded systems development activities are conceptualised as taking place within practices, and a concept for learning within practices is developed.

2.1 Embedded systems development

Embedded systems as defined in the introduction are used in a range of different contexts and functions from safety critical to infotainment [10], [22], [23]. Typical steps, carried out more or less in an iterative manner, would be systems specifications, systems modelling, integration, design optimizations, tests and prototypes [10], [11], [22].

Developers of the application SW are often separated organisationally and spatially from the other embedded system developers, who develop lower system layers, i.e. HW [3]. This is in stark contradiction to one main characteristic of the systems development, the up-down interfaces: the I/O devices interface with the HW level, yet the controlling interface mechanisms are built into the SW [3].

The systems development would often be composite, consisting of a mix of cross-functional and inter-organisational collaboration. Oshana and Kraeling [10] describe the inter-organisational communications needed for the collaboration as a principal responsibility of program management. Cross-functional communications must on the other hand be made to the program sponsor, senior management, the core team, and also extend this to external stakeholders [10]. Such communications should, according to Oshana and Kraeling [10], include verbal and nonverbal forms, face to face, to audiences and individually, with managers and with individual contributors, with local and international teams, also remotely and virtually [10].

2.2 Learning in practices

Embedded systems developers do systems design within unfolding practices. Drawing on Dewey [9, p. 32-34], the practice unfolds as consequences of the reciprocal interaction involving developers and the perceivable constitutive means, which entails that the social and the technical matters fuse into a sociotechnical composition [19], [20]; constitutive means are for instance a meeting room provided with different technical equipment, IT systems, video conference facilities and furniture or a production facility, as well as accessibility to sketches, drawings and physical breaker panels.

This research departs from conceptualising practice as an assemblage of matters [17] and especially the viewpoint that all involved developers comprise a homogeneous group. Given that developers have different faculties for sensing, feeling, thinking and doing, the composition of the development practices involves a dual heterogeneity. This means that we understand the practices to be constantly mutable in the reciprocal interaction between social and technical matters, and in addition, the system developers are neither passive individuals nor a homogeneous crowd. Rather the HW and SW developers are heterogeneous, and they demonstrate different levels of commitment and experience when specifying HW, SW and interfaces.

As a process, learning is defined as the transformation of an indeterminate situation into a determinate situation; a successful inquiry [9, p. 27]). An indeterminate situation arises due to disturbance in the developer's experience. Given that the developer's experience is inseparable from the unfolding practice, the indeterminacy arises in a situational composition of the practice. Thus, the learning process is enabled by the developer's experience and the usability of the perceivable constitutive means within the composition of the practice. A restoration of determinacy creates new experience for the systems developer. So saying, the outcome of the learning process is defined as new experience for the developer. The developer and the practice, however, are evolving in reciprocity [9].

Focusing on how the *doings* unfold within the specific composition of the practice makes it possible to grasp the reciprocity between the constitutive means and the developers' experience and thereby analyse the learning process. The *doing* is initiated when a developer is confronted with an indeterminate situation and strives to understand "what is going on here" [21, p. 8]; Dewey [9, chapter 6] defines this as the precognitive phase of the learning process. Drawing on Dewey [9, p. 112], an indeterminate situation grows out of an empirically real situation (like [21]), and to transcend the precognitive phase and thereby active reflective experience, it is crucial to ensure a proper anchoring of the indeterminacy. While a proper anchoring of the indeterminate situation paves the way for the individual's learning process, the prerequisite for a collective learning process is a converging anchoring of the indeterminate situation among the actively involved developers. Practising doings without a well-defined and empirically anchored problem implies that the developer(s) merely fumble through the learning process [9].

To analyse the embedded systems development and the learning process, we draw on Goffman's [21] strip of doings, which entails that a number of sequential doings gradually transforms the indeterminate situation into a determinate situation. For instance, two embedded systems developers face an indeterminate situation regarding the emergency stop of the wind turbine, which initiates the strip of doings. The strip of doings that follow are reading standards and specifications, dialogue on how the pitch system functions, drawing sketches on blackboard/paper, which gradually pave the way for clarifying the specification of HW, SW and interfaces, thereby achieving a determinate situation. Each one of these doings can in principle be treated as a topic to be subjected to an analysis [21, p. 564]. Not all doings lead to development and learning. Some doings are blocked and do not lead to a determinate situation.

This research subscribes to the idea that technology-as-text is inseparable from the systems developers' learning process. Likewise, the technology-as-text metaphor equates the development of embedded systems with the creation of text and is dependent on reading other texts. The use of the technology-as-text metaphor in this research is slightly different than the approaches applied by Grint & Woolgar [13], Latour [16], and Bijker [19], since it is here embedded in a pragmatist understanding of learning and agency [9].

Reading and writing doings draw on the reciprocal interaction between each of the involved SW- and HW-developers' experience and the perceivable constitutive means within the practice. Besides the reciprocity between a developer's experience and constitutive means, the verbal and non-verbal interplay among the actively involved developers either enables or constrains the transformation process of the indeterminate situation into the determinate situation.

3 Method

The understanding of the embedded systems development process is interpretivist [24], [25]. Within this broad strand, our approach builds on and extends an American Deweyan pragmatism conceptualisation of learning and the concepts of practices [17].

The case analysed below was selected from among four cases developed by one of the authors [26]. These four cases drew on an ethnographic field study and involved the researcher's presence at the embedded systems development project organisation three days a week in order to carry out observations and interviews, study written material and participate in meetings.

The case applied in this paper was selected because of its social and material complexity, and it encompasses both internal and inter-organisational development and learning processes that were either successfully conducted or ran into obstacles. The remaining three cases in the original study (which are not included in this paper) form the basis for the cross-case analysis in the discussion section, which methodically draws on Stake [27], meaning that each case is viewed as representing important and potentially unique insights about development and learning processes. It is assumed that the variations in the cases studied will provide insight into the complexity of development and learning processes; however, we do not claim generalizability.

This paper was written through revisiting the empirical material developed in the PhD by J.B. Mathiasen [26]. The original material relies on the same theoretical framework and focuses on product development in order to analyse SW and HW engineers' learning. Here, the understanding of the process shifts to viewing it as embedded systems development of the HW, SW and physical design that was carried out. We understand the shift from product development to systems development as a minor change in emphasis within the same research paradigm and methodological approach. It does mean, moreover, that the educational background of the embedded software developers is engineering (see for example [26]).

More specifically, the case material is collected in the following way. Over a period of one year, Hansen, the wind turbine control system manufacturer, was visited three days a week by one author. The researcher was present in an open-plan office of systems developers, with whom he engaged in informal conversation . Fourteen unstructured interviews, all documented with notes in Word documents, and 16 semi-structured interviews, all taped and transcribed, were carried out – 14 unstructured interviews in the beginning, and 16 semi-structured interviews at the end of the visiting period. In addition, documents were analysed and meeting attendance as well. As for internal meetings at Hansen, and inter-organisational project meetings, 56 were observed, each lasting an average of 95 minutes. The researcher adopted the role as passive observer [28]. The observations were recorded directly through notes in Word documents, and they dealt with the developers' dialogues and body language, how they used artefacts, cellphones, laptops and the blackboard, the characteristics of the room, whether or not the IT network was functioning, whether or not a person was reachable by cellphone etc. All quotes were translated from Danish to English by the authors.

By gradually juxtaposing the observations/interviews with our theoretical understanding, data were coded. The coding reveals two topics to be addressed in the analyses: the development trajectory and the contextual settings (like [29]). A micro-sociological approach, facilitating an analysis of each single doing as either a "reading doing" or a "writing doing", is applied to analyze the coded data. Therefore, conducting a development activity is considered to be a strip of doings [21]. By focusing on how the doings unfold, it becomes possible to grasp the reciprocity between a developer's experience and the constitutive means.

The limitations of the present research derive from the limited insight that a qualitative study focusing on individual systems developers gives regarding whether the development and learning processes found would occur in other embedded systems or other systems development processes. Oshana and Kaetling [10] contend that these types converge as embedded software and systems development becomes more complex, yet the present study is not able to contribute to this posit.

4 The wind turbine control system

The context for the selected embedded systems development case consists of the customer company (GlobeCorp), two consulting companies (M-Consult and G-Consult) and the specifying and system-producing company (Hansen). The customer for the system development project is a major corporation with headquarters and departments in Asia. It is represented in Europe through a department in Germany that focuses on renewable energy. The company developed its first complete wind turbine roughly ten years ago and a second six years ago. Neither were successes, and the Hansen representatives therefore view the customer as having limited experience with wind turbines. The two consulting organisations are specialists in mechanical engineering and gearbox respectively, and were therefore involved in the design of the new wind turbine to ensure mechanical stability with respect to the important issue of vibrations. Hansen's product, the embedded systems, regulates the other

components/systems, which entails that this product's functionalities will influence vibrations in nearly all components/systems in the wind turbine. GlobeCorp therefore made use of Hansen's experience regarding development of embedded systems (interview, SW developer 1). Hansen's experience was viewed as important for the entire turbine system.

The project organisation for the wind turbine development project encompassed a (customer) project group of 22 staff members in Germany. Hansen also established its own project group in Denmark, including its production facilities in Poland. As for the customer's project group, the collaboration with M-Consult and G-Consult was rather dense and focused on engineering services. In contrast, Hansen supplied physical breaker panels, an operational wind turbine control that integrates HW, SW and physical design, the aim being cost effectiveness and reliability. These different roles gave rise to conflicting perceptions of the task to be handled by Hansen.

According to the brief, the order commissioned consisted of parts of the embedded systems as HW and SW elements: the pitch control system, the converter, eight so-called breaker panels, one park control and monitoring system, a slipping ring (mechanical piece), cabling and aviation lights for the turbines. We focus here on the embedded systems, which means that the park control, slipping ring, cabling, and aviation lights are not discussed further until the cross-case discussion. At the outset of the development, the elements were all to be built according to well-known standard technical solutions.

Just after Hansen received the approved brief, the project group commenced clarifying the task. The brief was labeled a "mini technical specification" (miniTS), which consisted of 54 dense pages. According to both HW and SW developers, the miniTS was mainly a description of interface matters rather than an elaboration of the required functionalities.

Despite the fact that the miniTS had been approved by both GlobeCorp's management and Hansen's technical sales employees and management, the approach adopted was first to interpret the content of the miniTS, and second to develop the detailed specification. The initial interpretation of the miniTS involved interplay with the technical sales representative.

A sales representative presented selected pages of the miniTS on a TV screen and used the blackboard to elaborate his interpretation of the agreement with GlobeCorp. During the presentation, the project manager asked the salesman, "What have you sold? Is it a standard or a customised solution?" The discussion in the meeting room clearly revealed some disagreement among the employees involved. While the salesmen maintained that the order "is clear", HW developer 1 immediately identified several possible paths to follow: either to do a hard core specification or to keep the delivered documents at a high level and see what the customer will accept. The level of detail of both interface issues and acceptance tests were the core of this debate.

The interface issues caused doubts. Hence, the project manager arranged a meeting with the senior manager for the development department; the meeting took place in the open-plan office where the project manager and all developers were sitting together. Prior to the meeting, the developers drew up a comprehensive sketch of the HW solutions at the blackboard. After a while the senior manager said, "This sketch does only address the HW parts of the solution, but what are we going to do with the SW – what do you say, Tim?" (Tim is SW developer 1.) Until then, SW developer 1 had been rather mute, but he now replied, "If we lead the pen when developing the solutions instead of asking external suppliers to develop the HW solutions, we will have the opportunity to specify all SW by ourselves. As you can see all HW parts we have specified at the blackboard draw on well-known solutions and thus fit with our current SW solutions. If we chose other HW parts, we have to start from scratch."

At a later meeting with the customer, the interfaces between HW and the SW applications of the embedded systems came in focus again. First SW developer 1, and later on SW developer 2, presented the intended HW/SW interfaces to the Pitch systems (Pitch regulates the three blades of the wind turbine in accordance with the actual direction of wind and wind velocity and thus influences the mechanical stability of the wind turbine), and especially when to connect/disconnect the wind turbine to/from the grid. Following this, the customer representative went to the blackboard and made a comprehensive drawing of how they expected to handle mechanical issues related to the pitch and ideas to calculate the limits for connecting/disconnecting to the grid; the latter should be applied as guidelines for specifying HW/SW interfaces and for developing the SW applications to control the pitch and also the converter; Hansen purchased the pitch and converter from suppliers. The focal point for the dialogue gradually turned out to be a sensor, which would be activated, when a mechanical part reached predefined limits. Hansen needed information from GlobeCorp regarding the length of the necessary movement and the rotation (wind) speeds when connecting/disconnecting the wind turbine; this kind of information/specifications should be embedded into the SW solution, which also had an influence on the detailed development of HW solutions.

For instance: HW developer 1 asks the customer representative, "Why do you have different maximum speed on the generator?" Customer representative: "I cannot see it. Oh now I can see it… (he walks to the blackboard)… The upper limit is where we disconnect the wind turbine, an emergency stop. First, we disconnect the generator and the converter, whereafter we pitch the blades." HW developer 1: "Yes, I understand you, but I cannot

understand why you disconnect the converter and generator. Our idea with an emergency stop is to have the generator and converter working as long as possible and thereby help to brake the blades…. We need to define the minimum and maximum generator speed." The customer representative did not understand why this information was crucial for Hansen, and again he emphasised that Hansen should teach him and his developers how all their HW and SW solutions function – otherwise, he/they would be incapable of contributing to the specification of the HW and SW solutions. Later on, HW developer 1 raised issues regarding the yaw system. The customer representative explained, "We have complex terrain in X (home country of GlobeCorp), where the wind changes suddenly. We have been working on solving/finding a solution regarding yawing the wind turbine. We would like Hansen to come up with a proposal for a solution." Again, the customer representative put forward that he needed to understand the functionalities; otherwise, he could not contribute with relevant information. HW developer 1: "I don't know. I have to ask our SW-employees about this." The HW developer makes a note: "GlobeCorp requests a yaw strategy for complex territories."

In any case, at several subsequent meetings between Hansen and GlobeCorp, there continued to be an "arm's length" situation, where the two actors did not quite want the same. The tendency was that Hansen's developers did the specification internally rather than in close collaboration with the customer, while GlobeCorp wanted to enhance the learning about HW and SW solutions.

Later on, GlobeCorp initiated a large coordinating meeting between Hansen, M-Consult and G-Consult to make sure that all coordination about interfaces was in place. To optimise the mechanical stability, some collaboration between the HW/SW developers and the two consulting organisations was necessary. In this regard, M-Consult conducted lots of calculations and simulations concerning load and stress on the components in the wind turbine. SW developer 3 explained, "In order to carry these out, they need to know how we (Hansen) regulate the wind turbine. For instance, depending on how we pitch the blades in the wind, you will get different forces and vibrations in the components. And as the gearbox is normally a problem, you have to minimise these forces and vibrations."

The meeting turned out to be flooded with customer representatives. It turned out that the important dialogue was between Hansen, M-Consult and G-Consult, whereas the various customer representatives responsible for mechanics, gearbox, electricity and hydraulics eagerly took notes. In general, to the Hansen representatives, the meeting meant that they were assured that all issues actually were solved, yet optimisation options still remained. SW developer 1 told it this way: "... I went to a meeting yesterday, where these gear people were present, and I inquired into the issue and got some information. And now I am going to sit down and describe how gear cooling and the lubrication system can work, and I will then develop a control strategy, where I describe how I want to control it, and this I will send directly to G-Consult and ask them to provide comments. So here we return with something different from the pre-clarification [miniTS] in the order."

As time went by and the time pressure became critical, yet still more customer demands emerged: "... Our panels should come from Hansen in 14 days. At this moment, they are sitting and analysing an old gearbox (G-Consult), trying to find out which design changes are needed. They have another ten days and then the gearbox needs to be manufactured and tested; otherwise, the turbine is due in another three months. Now, new issues are arising, however, because these gear people – they talked purely mechanical design, and I began talking lubrication strategy and cooling strategy. That made them sit and think, and then I told them that sometimes a turbine rewinds; it goes the wrong way and some gearboxes do not like that…" (interview, SW developer 1).

SW developer 1 further unfolded issues with choices of mechanical parts or electrical pumps in the light of grid breakdowns, illustrating the incomplete interplay that was unravelling as well as coordination issues with other companies. For instance, the brake system involved a sub-supplier and the gearbox manufacturer belonged to another division of GlobeCorp. Hansen also struggled with keeping HW and SW coordinated with the mechanical design: "It is crucial to coordinate the I/Os, because we [HW and SW developers] are very dependent on each other. If the HW is far ahead of the SW development, it often becomes problematic to coordinate crosswise…, but if we do not know the HW – for instance, the gearbox – we need information in this regard. Or if modifying the HW, it is crucial to coordinate these changes with the SW." (Interview, SW developer 1.)

Accordingly, a rather direct jump from 3D drawings and electrical connection diagrams to a physical prototype had to be made. Also, the documentation was developed to describe exactly how the prototype wind turbine should be operated, with which responsibilities for Hansen and which for GlobeCorp. The prototype embedded systems were eventually delivered and tested, and only slightly delayed.

5 Analysis and discussion

In the following, we commence by analysing the embedded systems development and learning processes in the cross-functional, daily, and inter-organisational practices and its commercial context (5.1). Then, we continue the analysis by comparing across practices.

5.1 Learning processes in context

Hansen's developers have a clear understanding of GlobeCorp's requirements/wishes during the preparation of the detailed technical specifications. However, this understanding is not in accordance with the pre-clarifications. This discrepancy emerges gradually, starting in the internal cross-functional Hansen collaboration involving management, sales and the project group. Later on, it emerges into the inter-organisational collaboration between GlobeCorp and Hansen. Some of Hansen's representatives have their own strategy of keeping technical knowledge close to their chests, and Hansen's developers have many years of experience. These features together permeate the strips of doings and result in Hansen's developers taking charge and thus handling the detailed specifications phase. The discrepancy influences the learning processes differently.

With regard to the *cross-functional practices* (management, sales and the project group), the pre-clarification document is intended to function as a constitutive means to ensure an alignment of all HW, SW and mechanical aspects to be developed for GlobeCorp. Within these practices, the pre-clarification document is pivotal for the reading doings, and also to a large extent for the verbal and non-verbal interplay among the involved employees. This interplay, concurrent with on-going reading doings of the perceivable constitutive means, disturbs the HW and SW developers' understanding, paving the way for transcending the precognitive phase. This proper anchoring of the indeterminacy enables the reflective experience. However, due the fact that the two groups of employees (management/sales versus HW/SW developers) have different experience in relation to embedded systems development, the perceivable constitutive means guide their reflective experience differently. Put briefly, the HW/SW developers' experience is far more in-depth and detailed. As the involved employees' reflections follow divergent tracks, the strip of doings is constrained, implying that the embedded systems development and learning processes run astray. This means that the composition of these practices, involving sales and HW and SW developers, enables neither individual nor collective learning.

Learning does occur in the *daily practices* of embedded systems development involving Hansen's project manager and HW, SW and other developers. Given that Hansen's project manager and all involved developers are placed together in an open-plan office, the composition of the practices is highly mutable: the blackboard is overfilled with sketches; key components from suppliers are placed on the tables; and developers' reading doings include various standards, drawings, electrical diagrams, and bill of materials. This diversity of perceivable constitutive means, the use of laptops to retrieve information, reading e-mails aloud as well as the verbal and non-verbal interplay among HW and SW developers are crucial for ensuring a convergent anchoring of the indeterminate situation.

If Hansen's developers lack information, however, to either facilitate a continuation of the strip of doings or to achieve a convergent anchoring of the indeterminacy, they submit questionnaires to GlobeCorp's project manager in an attempt to gain access to the necessary information. If GlobeCorp is incapable of handing over the requested information, the HW/SW developers deliberately orchestrate this as an indeterminate situation at the following inter-organisational meeting with GlobeCorp. All preparatory work (strips of doings) to ensure the orchestration of the indeterminacy within the inter-organisational practices is only accomplished within the daily practice; this development approach influences the inter-organisational learning (see below).

In general, the HW/SW and other developers' collaboration within the daily practices makes an effort to ensure a convergent anchoring of the indeterminacy. If crucial information is lacking, it is possible for the developer(s) to ask one of the neighbouring developers or to block deliberately the strip(s) of doing until the required information has been retrieved.

Thus, the daily practice composition is characterised by mutability and a high degree of usability of diverse constitutive means to guide the developers' reflections. In combination with the interplay among the developers, this enables a convergent anchoring and also a continuation of the strips of doings – i.e., learning internally in Hansen's project group.

As mentioned above, within the *inter-organisational practices*, the HW/SW developers have prepared (within the daily practices) the orchestration of the indeterminate situation (deliberately anchoring the task to be handled) so as to gain access to crucial information; thereby, representatives from GlobeCorp hand over dedicated information, and by gaining access to this information, Hansen's developers are capable of drawing up the detailed

specifications within the daily practice. Hence, in relation to the preparation of the detailed specifications, none of the strips of doings are collectively conducted within the inter-organisational practices; even if these practices involve Hansen's developers as well as GlobeCorp's developers. The analysis reveals three issues in relation to this lack of collective learning:

First, only GlobeCorp's project manager participated in the drawing up of the miniTS document. In this respect, the transition from the miniTS to the detailed specification phase is challenging and results in a rejection of the miniTS document; consequently, the constitutive means become unusable. As a result of this rejection, a great many strips of doing are blocked within the inter-organisational practices; instead, they should have invested resources in achieving a proper and collective anchoring of the indeterminacy and guiding the developers' reflective experience.

Second, the inappropriate constitutive means combined with the aforementioned protection of technical knowledge – and thus how transparent explanations of technical issues should be – leaves GlobeCorp's developers with nothing to guide their reflections, apart from the memories of the miniTS phase. This means that the two groups of developers' reflective thinking did not follow a convergent track. While GlobeCorp's reflections addressed an eagerness to gain detailed knowledge of the technical solutions, the Hansen developers' reflections focused on gaining access to necessary information.

Third, from time to time the interplay between GlobeCorp's and Hansen's developers borders on a conflict, which entails a divergent anchoring of the indeterminacy. The dialogue, verbal and non-verbal, among the developers illustrates that each of the two groups of developers draws on their reflective experience to stick to their own position. In line with this, given that the developers are incapable of achieving a convergent anchoring of the indeterminate situation to be handled, the reflective experience of both groups of developers follows divergent tracks. This means that both the embedded systems development and the learning processes are constrained and finally run astray.

Hence, within the above inter-organisational practices, the preparation of the HW/SW and interface specifications does not draw on reciprocity between each of the involved developer's reflective experience and perceivable constitutive means. Determinants for derailing the collective learning processes are the unusable constitutive means as well as the tendency to protect technical knowledge among the involved developers, with the result that they are incapable of achieving a common ground in terms of the indeterminate situation to be handled. Instead, the preparation of the detailed specifications is the outcome of Hansen's developers merely handing over and retrieving information. Hence, successful learning occurs within the daily practices, but not in the inter-organisational practices.

Regarding the unfolding *inter-organisational practices* to ensure an alignment of HW/SW issues and interfaces to other crucial sub-systems and components in the wind turbine, the composition of these inter-organisational practices is adapted to facilitate a converging anchoring and a continuation of the strips of doings. Heterogeneous developers from Hansen, GlobeCorp and the two consultant companies, M-Consult and G-Consult, draw on various constitutive means and utilise their different reflective experience to ensure successful strips of doings. Through these strips of doings, GlobeCorp's position becomes clearer.

The perceivable constitutive means within this practice reveal different characteristics regarding the levels of detail, yet the involved developers could apply them to reach common ground with regard to the substance of the indeterminacy to be handled. Likewise, the developers from the two consultant companies and from Hansen have great amounts of experience within embedded systems development and especially in relation to their own technical field(s); some have crucial experience related to pitching the blades, others gearboxes, lubrication etc. Despite these different specialist backgrounds, the interplay among the developers makes it possible to achieve a convergent anchoring of the indeterminacy to be handled – for instance, issues arising if a wind turbine should rotate in reverse, which some gearboxes do not like. The anchoring of the indeterminacy addresses a specific interface issue to be handled, and the case reveals a time-wise dependency between the development of HW and SW development. To guide their reflective experience, the involved developers either draw on 'prefabricated' constitutive means or make sketches. Leaning thus occurs within the composition of this inter-organisational practice.

5.2 Discussion across practices

The case discussed above relates to one customer, GlobeCorp. Nevertheless, the reciprocity between a developer's experience, constitutive means and the verbal/non-verbal interplay among the developers unfolds in characteristic ways. The embedded systems case becomes complex and even political. The analysis illustrates complex "experience-constitutive means", reciprocity and interplay among developers, who have different kinds of specialist experience in HW, SW, mechanics and production. The case analysed reveals even more complex reciprocity and interplay in facilitating the embedded systems development and learning processes than could be

expected in an embedded systems case. In the setting of delivering to GlobeCorp, Hansen has the opportunity to move the boundary between the wind turbine manufacturer, the purchaser and Hansen's own services, and enter an area of combined SW and HW systems development. The customer, GlobeCorp, wishes to create an environment for mutual learning [2], yet for various reasons, Hansen does not play along, which implies that practices constrain the inter-organisational learning and combined systems development.

The composition of practices, the anchoring of the initial indeterminate situation, and the continuation of strips of doings reveal some interesting patterns.

Across the practices, the learning enabling elements encompass high accessibility to constitutive means and different levels and character of experience among the developers. If the constitutive means, such as a specification, is too malleable [20], it will create insecurity, whereas if it is too obdurate, our findings reveal that the risk exists that any learning and embedded systems development will be blocked.

The process of anchoring is crucial for the learning process. A proper anchoring of the indeterminacy enables transcending the precognitive phase, thus paving the way for the system developers' reflections. The enabler for a continuation of the individual's learning process is the reciprocity involving the system developer's reflective experience and the constitutive means. However, collective learning calls for a convergent anchoring among the involved developers and subsequently both reciprocal interaction between the constitutive means and the developers' experience and verbal/non-verbal interplay. The analysis reveals that the anchoring varies across inter-organisational, cross-functional, and daily practices. But one type of practice is particularly interesting, because of its winding and resource-demanding trajectory. This learning trajectory is nevertheless characterised by high accessibility to diverse and usable constitutive means as well as developers with different levels of experience and specialist background. This enables the reciprocity between the constitutive means and the developers' experience.

The practices characterising embedded systems development are thus constituted by a larger heterogeneity than other systems development practices. This is due to the combination of SW, HW and interface aspects. The heterogeneity is thus double – due to different areas of needed development and due to different individual developers involved.

Analytical frameworks to understand practice-based learning (among others, [5], [6], [7], [30]) normally do acknowledge a situated working practice. In general, however, these analytical frameworks to study practice-based learning consider individuals as being a homogenous crowd having similar learning preferences and motivation for action; being involved in the practice enables learning *per se*. The analytical framework in our research acknowledges dual heterogeneity: we understand the practice to be constantly mutable, and we understand that embedded system developers are heterogeneous and that they demonstrate different levels of commitment when developing HW and SW. Consequently, the analytical framework focuses attention on how an activity – a strip of doings – unfolds within a practice, making it possible to grasp that both the developers and the practices are heterogeneous and that developers and practice are evolving in reciprocity. By doing so, the analytical framework paves the way for grasping and thus analysing the characteristics that enable or constrain the learning process when conducting an embedded systems development activity.

One implication for embedded systems development is that the often advocated intimate relationship with the customer representatives [10] might be overruled by supplier strategies for keeping customers at an arm's length in order to maintain and protect know how. To enable the complex set of diverse knowledge and developer skills for embedded systems development, managers should ensure that constitutive means, such as specifications, are available and that they are sufficiently obdurate. Too malleable means, such as customers with unclear demands, can constrain both the learning processes and embedded systems development. Systems developers with different experience also enable learning processes. Converging anchoring of the indeterminate situation being handled enables collective learning. Focusing on this, systems development companies can safeguard their learning and product embedded knowledge when engaging in inter-organisational collaboration where they must handle the risk of giving knowledge away.

6 Conclusion

The aim of this paper is to investigate the learning that occurs in an embedded systems development project of a wind turbine control system, and to inquire into the enablers and constraints the development process features. Focusing on micro-trajectories of learning processes, the framework of understanding highlights the importance of moving from an indeterminate to a determinate situation through strips of doings, when developing SW, HW and mechanical components and integrating them into the embedded system. The analysis shows how learning can occur within different practices throughout embedded systems development, if the developers are actively involved in handling the transformation from the indeterminate to determinate situation.

In the case studied, one constraint is the striving for cost efficiency and swift development, which can lead to processes that cause learning to run into obstacles. Also the protection of existing knowledge becomes a constraint. Neither individual nor organisational learning occurs automatically during embedded systems development, but central enablers for learning are heterogeneity amongst systems developers, successful anchoring of indeterminacy, and the composition of the practices. The practices characterising embedded systems development are thus constituted by a larger heterogeneity than other systems development practices, due to the combination of SW, HW, interface and mechanical aspects. The heterogeneity encompasses both the different areas of development needed and the different individual developers involved.

References

1. Technavio: Global Embedded Software Market 2015-2019. Technavio. Downloaded from http://www.technavio.com, April 2016 (2015)
2. Finnegan P., Galliers R.D., Powell P.: Applying triple loop learning to planning electronic trading systems, Information Technology & People, vol. 16 iss. 4, pp. 461 – 483 (2003)
3. Vidgen R, Wang X.: Coevolving systems and the organization of agile software development". Information Systems Research, vol. 20, no. 3, pp. 355–376 (2009)
4. Lyytinen K., Rose G.M.: Information system development agility as organizational learning. European Journal of Information Systems, vol. 15, pp. 183–199 (2006)
5. Brown, J., Duguid P.: Organizational Learning and Communities-of-Practice: Toward a Unified View of Working, Learning, and Innovation". Organization Science, vol. 2, no. 1, pp. 40–57 (1991)
6. Gherardi, S., Nicolini, D.: To Transfer is to Transform: The Circulation of Safety knowledge. Organization, vol. 7, no 2, pp. 329-348 (2000)
7. Nonaka, I., Toyama, R., Konno, N.: SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planning, vol. 33, pp. 5-34 (2000)
8. Dewey, J.: Essays and How We Think. Revised Edition, Southern Illinois University Press, The Later Works Volume 8. Illinois (1933)
9. Dewey, J.: Logic: The Theory of Inquiry. Southern Illinois University Press, The Later Works Volume 12. Illinois (1938)
10. Oshana R., Kraeling M. (eds.): Software Engineering for Embedded Systems: Methods, Practical Techniques and Applications. Newnes, Waltham (2013)
11. Bianchi, F.D., Battista, H., Mantz, R.J.: Wind Turbine Control Systems. Principles, Modelling and Gain Scheduling Design. Springer. Berlin (2007)
12. Tjornehoj G., Mathiassen L.: Between control and drift: negotiating improvement in a small software firm. Information Technology & People, vol. 21 no. 1, pp. 69 – 90 (2008)
13. Grint, K., Woolgar, S.: The Machine at Work, Technology, Work and Organisation. Cambridge Polity Press. Cambridge (1997)
14. Henderson, K.: The Role of Material Objects in the Design Process: A Comparison of Two Design Cultures and How They Contend with Automation. Science, Technology, & Human Values, vol. 23, no. 2, pp. 139-174 (1998)
15. Hutchby, I.: Technologies, Text and Affordances. Sociology, vol. 35, no 2, pp. 441-456 (2001)
16. Latour, B.: Where Are the Missing Masses? The Sociology of a Few Mundane Artifacts. In: W. E. Bijker, J. Law. (eds.) Shaping Technology/Building Society: Studies in Sociotechnical Change. The MIT Press Cambridge, pp. 225-258 (1992)
17. Orlikowski, W.J., Scott, S.V.: Sociomateriality: Challenging the separation of technology, work and organization. Annals of the Academy of Management, vol. 2, pp. 433–474 (2008)
18. Akrich, M.: The De-Scription of Technical Objects. In: W.E. Bijker & J. Law. (eds.) Shaping Technology/Building Society: Studies in Sociotechnical Change, pp. 205-224. The MIT Press Cambridge (1992)
19. Bijker, W.E. How is technology made?—That is the question!, *Cambridge Journal of Economics*, vol. 34, 63-76 (2010)
20. Henderson, K.: On Line and On Paper. Visual Representations, Visual Culture, and Computer Graphics in Design Engineering. MIT Press, Cambridge, MA. (1999)
21. Goffman, E.: Frame Analysis. Penguin Books Ltd. Harmondsworth (1974)
22. Suri N., Jhumka A., Hiller, M., Pataricza, A., Islam S., Sârbu C.: A software integration approach for designing and assessing dependable embedded systems. Journal of Systems and Software, vol. 83, iss. 10, pp. 1780-1800 (2010)
23. Yu T., Sung A., Srisaan W., Rothermel, G.: An approach to testing commercial embedded systems. Journal of Systems and Software, vol. 88, pp. 207-230 (2014)
24. Howcroft D., Trauth E.: Handbook of Critical Information Systems Research -Theory and Application. Edward Elgar Publishing. London (2005)
25. Walsham G.: Interpreting Information Systems in Organizations. Wiley. Chichester (1993)
26. Mathiasen J.B.: Learning Within a Product Development Working Practice: An Understanding Anchored in Pragmatism. PhD diss., Copenhagen Business School (2012)
27. Stake, R.E.: Case studies. In: N.K. Denzin, Y.S. Lincoln (eds.). Handbook of qualitative research, 2nd ed. Thousand Oaks: Saga Publication Inc. Thousand Oaks, pp. 435-454 (2000)
28. Bryman A, Bell E.: Business Research Methods. 3rd Ed. Oxford University Press. Oxford (2011)
29. Elkjær B.: Organizational Learning: The 'Third Way'. Management Learning, vol. 35, no. 4, pp. 419-434 (2004)
30. Carlile, P.R.: Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries. Organization Science, vol. 15, no. 5, pp. 555-568 (2004)