Archetypical Approaches of Fast Software Development and Slow Embedded Projects

Ulrik Eklund
Malmö University
School of Technology, Dept. Computer Science
Malmö, Sweden
Email: ulrik.eklund@mah.se

Jan Bosch
Chalmers University of Technology
Dept. Computer Science and Engineering
Göteborg, Sweden
Email: jan.bosch@chalmers.se

***Abstract*—This paper describes the problem context of software development for mass-produced embedded systems, with distinguishing factors such as the co-design of software and hardware, strong focus on manufacturing aspects, supplier involvement and safety-critical functionality. In this context there is a need for a *holistic model* to explain the failures and successes industrial projects, where just investigating a single dimension, e.g. chosen ways-of-working or architecture is not sufficient.**

The main contribution is a holistic model consisting of five archetypical approaches to embedded software development, based on a mapping study over industrial cases in literature. The approaches range from "traditional" stage-gate projects focusing on product qualities and large integration efforts, to fast development in short loops by autonomous teams based on a composable software platform. The model aligns the processes with the architecture of the embedded software, and the implications on the business and the organisation. The model allows an research & development(R&D) organisation to identify where it is positioned and to evolve its software development approach. The model is elucidated by two empirical cases from a Swedish company.

***Keywords*-embedded software; software engineering; software architecture; business; companies**

I. Introduction

Software is prevalent in many products manufactured today; cars, washing machines, mobile phones, airplanes and satellites [1]. Typically these products are developed in large and complex industrial projects where the embedded software may be critical for the success of the product but the manufacturing and delivery of the product is a heavier investment than the R&D budget. This in turn tends to drive the entire R&D process, and software just follows the process logic of the manufacturing setup.

Ebert & Jones [1] mentions factors contributing to the complexity in their survey of the state of embedded software development: "combined software/hardware systems equipped with distributed software, computers, sensors, and actuators" which points to the integration aspects of these systems. They list "high demands on availability, safety, information security, and interoperability" as typical quality attributes. Liggesmeyer & Trapp [2] reaffirms this view stating that embedded software is one of many elements in a product consisting of mechanics, electrics and electronics, and software. The also mention quality attributes such as software safety, reliability and timeliness software safety, reliability, and timeliness, which all must taken into account in the development process. Manhart & Schneider [3] describe agile development of software in buses at Daimler-Chrysler. They mention for example that "equipment, functions, or parameter sets are implemented by integrating different proportions of third party- and OEM manufactured components" indicating supplier involvement.

Examples like this, together with our own research [4], allows us to describe the domain of large industrial development of mass-produced embedded systems (MPES) by five characteristics:

- Deep integration between hardware and software for significant parts of the functionality
- Strong focus on manufacturing aspects of the product in the development
- Strong supplier involvement
- Some parts realise safety-critical functionality
- Long production life-time

In our previous research of MPES development there were significant characteristics of the studied cases that could not be explained only by looking at the used development process or chosen architecture [5]–[7]. For example in [5] there was a desire to simplify the working method and develop functionality top-down from a customer's point of view. But the architecture used made for cumbersome integration of subsystems, this together with the used development process "led to monster documentation". This in turn affected the ability to outsource module development, with suppliers having difficulty in handling and meeting the specifications, which affected cost negatively.

In the third case in [7] the goal was to share modules with other car manufacturers to get advantages of scale, and to outsource a significant part of module development. But the distributed architecture meant the interfaces between the modules were complex. This in combination with the stage-gate process used initially meant that the development effort was underestimated. Later in the project the shorter development iterations were introduced, which probably "saved" the project launch date.

The complexities of the studied cases establish the need for a *holistic model* to explain the failures and successes of these and other industrial projects. We propose a model, built by

analysing existing industrial approaches of embedded software through the four dimensions of Business, Architecture, Process and Organisation (BAPO). These four dimensions originate from the framework by van der Linden et al. [8], aimed at evaluating product families. The use of BAPO allows the construction of the model and identifies different approaches to development of MPES products.

The main contribution of this paper is a holistic model for aligning software development processes with the architecture of the embedded software, and the implications these have on the business and the organisation. The model allows an R&D organisation to identify where it is positioned and how to evolve its software development approach in terms of architecture, process and organisation. The model is illustrated through two industrial cases. In addition to this, the paper provides a rich insight into the context and challenges for development of industrial systems and their embedded software both by empirical evidence from first-hand industrial cases and by published literature through a mapping study.

II. Research Methodology and Problem

The paper builds an explanatory holistic model with the properties described in Section I, through a 3-stage research process. The first stage of the research process identifies the research questions to be investigated:

1) What approaches to embedded software development are used in industry?
   a) How do these approaches relate to used architectural styles?
   b) What business and organisational implications does it have for an organisation to develop software with a specific approach?
2) How would a model over the processes and architectural styles and their relationship to business and organisation look like?

The second stage, described in detail in Section III, is a mapping study identifying published industrial cases relevant to the research questions above.

The last stage of the process builds a qualitative model, described in detail in Section IV. The model articulates viable approaches an R&D organisation takes to embedded software development. The model is also elucidated through two industrial cases in Sections IV-A1 and IV-B1. These first-hand cases were captured at Volvo Car Corporation (VCC) in a qualitative manner. The case studies took advantage of the fact that the first author was native to the case company at the time (as defined by [9]), and acted as a participant/observer.

III. Mapping Study of Development Approaches

The mapping study surveyed existing literature with the aim to identify industrial cases relevant to the research question. The study was performed similarly to a systematic literature review [10], with a focus on the steps *emphasized* below:

1) Planning of the mapping study
   - Establish the need of a review
   - Define the search questions
   - Define the review protocol
   - Evaluate the protocol
2) Conducting the mapping study
   - *Identify relevant cases*
   - *Choose primary papers based on relevance in case description*
   - *Asses qualities of chosen papers*
   - *Extract relevant categories of development approaches and architectures from the chosen papers*
   - *Synthesise the results*
3) Reporting the mapping study

A. Define Search Questions

The aim was to identify and model the relationship between business, architecture, process and organisation when developing embedded software, and we iteratively refined the search queries until we ended up with the final phrase used in the Scopus article database.

```
 (TITLE-ABS-KEY(software) OR TITLE-ABS-KEY({S/W}))
AND
 (TITLE-ABS-KEY(embedded) OR TITLE-ABS-KEY(automotive))
AND
 (TITLE-ABS-KEY(architecture) OR TITLE-ABS-KEY(integrat*) OR
  TITLE-ABS-KEY(compos*) OR TITLE-ABS-KEY(platform))
AND
 (TITLE-ABS-KEY(agile) OR TITLE-ABS-KEY(process))
AND
 (TITLE-ABS-KEY(project) OR  TITLE-ABS-KEY(industr*) OR
  TITLE-ABS-KEY(compan*))
AND
 (TITLE-ABS-KEY(case stud*))
```

Search phrase 1: Search phrase used in Scopus.

We limited the search to the last ten years (2003-2012), and excluded everything not within the Subject Areas of *Computer Science*, *Engineering* or *Business, Management and Accounting*. This initial search resulted in 117 papers.

B. Review Protocol

From the list of 117 papers we applied the inclusion and exclusion criteria in Table I. These criteria were applied in two rounds, the first round by just reading the abstracts, which left 53 papers. The second round evaluated the full papers, and this resulted in 23 papers, of which the authors were involved in 4 papers [6], [11]–[13].

C. Quality Assessment

We were looking for examples of where an organisation develops a system with embedded software in an *industrial* context and the range of development approaches and different styles of architectures used therein. We therefore selected case studies as a search criterion since we wanted empirical studies "that investigates a contemporary phenomena within its real-life context..." [14]. A number of papers claimed to do a case study without providing any information about the context of development, e.g. the purpose of the system. Nor did they say anything about about the organisation doing the development, e.g. was it done by an industrial team, PhD students, or a mix of academic and industrial participants? We classified these papers as academic proof-of-concept prototypes, and therefore excluded according to EC1. These papers were a significant part of those excluded resulting in the final set of 23 papers.

Table I
USED INCLUSION & EXCLUSION CRITERIA.

Inclusion criteria	
IC1	Papers, technical reports, theses, industry white papers and presentations describing industrial software development approaches and architectures for embedded systems.
IC2	If a search result contained several cases each case were counted separately.
Exclusion criteria	
EC1	Studies that did not report on projects in an industrial context, e.g. student projects, open source communities, prototypes developed in academia, etc.
EC2	Studies only focusing on hardware development
EC3	Studies only evaluating specific tools, notations or other modelling techniques
EC4	Studies only evaluating testing or other verification practises, such as formal methods
EC5	Survey papers over other papers
EC6	Textbooks and proceeding summaries
EC7	Material not accessible either freely through the world wide web or through available library resources
EC8	Material not in English, Swedish or Dutch
EC9	If a case was published several times only the most recent was included

Our insider knowledge allowed us to identify when multiple papers described the same case, e.g. developing a new car year model on an existing platform at Volvo Car Corporation [6], [15]–[17], or the continuous evolution of a heavy vehicle product line [6], [18].

D. Study results and limitations

The final selection of papers resulted in 28 cases presented in 23 papers. Each case were categorised on how they addressed the dimensions of business, architecture, process and organisation (BAPO).

A limitation is the difficulty to asses if other, not found, industrial cases would expand or alter the resulting model. Another limitation is the scope of found cases, the search query finds cases which call themselves "embedded", but this defintion is probaly not uniform which affects the applicability of the resulting model across various domains. A third limitation is that few papers had information on all four categories when describing the cases.

Despite the limitations it is still possible to draw some general qualitative conclusions: The common way of working is to follow a sequential order of activities, which is characterised as a V-model [19] or as a stage-gate process [6], [20]. Some papers describe cases of agile development, either for a system as a whole [11] or by individual teams within the organisation [20]. When described, most case architectures focus on enabling product quality attributes, usually domain-specific such as safety, cost and variability in the automotive domain [13], security in defence [21], and dependability in space [22]. Some organisations utilise a product line architecture to enable tailoring of the system to a particular customer [23], [24]. An important architectural issue is the integration of modules or sub-systems to a working whole [6], [12], [18], [25].

We can conclude that there is no single approach of how software in embedded systems is developed, but there is rather a range of various approaches.

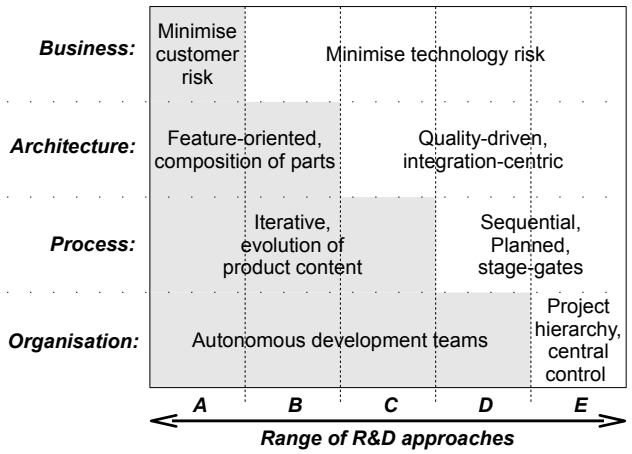

Figure 1. Model over the five identified archetypical R&D approaches when developing embedded software.

IV. APPROACHES TO DEVELOP EMBEDDED SYSTEMS

The last stage of the research process resulted in a model able to answer the research questions in Section II. By evaluating how the cases are aligned in the four dimensions of BAPO we identified *five archetypical R&D approaches* that are feasible within industrial contexts, archetypical in the sense they are the original pattern or model of which all development approaches of the same type are representations or copies. These five approaches constitute the model (Figure 1). The model is described with a narrative of each approach, starting from the right (E) since this is where most cases are. The 28 cases also imply an evolution of how an organisation can move along the range of the five R&D approaches, from right to left in our model (from E to A). We saw no case in the mapping study describing an organisation evolving in the opposite direction.

Primarily the model is explanatory in the sense it conveys how industrial projects align their concerns of business, architecture, process and organisation, thus is an empirical instance of the BAPO framework in [26]. Secondary, the model can be prescriptive in two ways; it suggest to an organisation how others align the BAPO concerns, and it suggest a path of evolution to alternative R&D approaches.

A. Approach E: Rorqual Organisations

Organisations using this approach run development projects demanding a lot of investment in technology during R&D, both what goes into the product and what technology is required to manufacture it. This can be considered the standard practice at which MPES software is developed, with six cases clearly falling into this category (described in papers [6], [13], [15]–[17], [19], [21], [27], [28]). An additional twelve cases (in papers [22], [23], [25], [29]–[31]) fall either into this category or category D.

Business: One major business driver is to minimise the risk associated with the technology investments. Software may not

even be considered as a major risk compared to e.g. hardware or manufacturing investments.

Architecture: The architectures and technology used for subsystems and their embedded software optimise the desired product requirements rather than being concerned with the difficulty for the organisation to develop and maintain them them. As a result much of the architecting effort is spent on integration issues.

Process: The process model used is a stage-gate model [32]. The project planning follows a template based on calendar time, which has evolved from experiences from previous projects. Gate progression corresponds to design artefacts, e.g. user requirements, system requirements, system & software architecture, component requirements, software implementation (i.e. code), and verification & validation, i.e. a V-model even if the artefacts can be updated as the project progresses.

Organisation: The development organisation is functionally structured, clustering domain specialists together. The organisation resembles a hub with spokes with a central systems engineering or architecture team responsible for the complete product properties, new or updated features are negotiated along the spokes before incorporated in the product. Coordination of involved teams is done through synchronisation of processes. It is common practice to outsource part of the electronics and software development to subcontractors.

1) Example: Development Project of an Infotainment System.: VCC decided to deliver a new generation of infotainment systems to extend its competitive position. The development organisation had to deal with several prerequisites which had a major effect on the project: The systems were to be sold by more than one brand within Ford Motor Company and some developed components were to be shared between brands (both hardware and software) while maintaining a brand-specific HMI. This was to leverage sourcing with other (unrelated) components from suppliers. There was also a desire to minimise the requirements elicitation effort in terms of spent man-hours. The project changed it setup midway since the initial development approach had problems delivering according to schedule. A post mortem analysis was done to identify the major factors influencing the architecture and the causal relationships between them and the used process. Management at the concerned department at VCC ordered the study to learn from this case. The case is previously published in [7].

Business: The main business driver to develop a completely new system was to "keep up" with the technological evolution and competitive pressure within the in-vehicle infotainment domain. The release of the new system coincided with the release of a new car model, the Volvo S60 to leverage marketing.

Architecture: The architecture changed, unintentionally, compared to the previous generation infotainment system. Some of the most complex customer features were distributed between two electronic control units (ECUs), which led to a complex interface shared between two software suppliers. The architecture was based on established technologies in the automotive domain, e.g. Media Oriented Systems Transport (MOST) [33] for communication buses, while the application interface on top of this did not follow standard MOST services. The main driver for the separation of HMI to one ECU while having the core functionality in another ECU was to allow sharing of the latter ECU between different car brands while allowing for a brand-specific HMI.

Process: The initial process approach was typical for waterfall development, with the software specifications for each ECU being reused from the previous generation to minimise the effort in writing new specifications. Initially the focus was on component development, i.e. on each ECU with its deployed software. The project progress was initially measured in implemented customer features, in spite of the component-focus, this meant that delivery of infrastructure software necessary for integration and testing was initially de-emphasized. It was difficult to plan and manage the integration occasions necessary for validation and verification since there was no overall view of feature realisation.

Organisation: All software was to be outsourced while in the previous system generation the software for the main ECU was developed in-house at VCC. This meant new development practices of working with supplier had to be established.

The project in this case started at the rightmost E position, but without performing some key elements common to successful projects in this position, e.g. a clear architecture. The project changed approach midway with shorter sprints with a limited set of features verified after each sprint rather than planning against large integrations. The setup of the teams changed, from being focused on component development to cross-functional teams focused on feature development. These changes were vital in keeping the launch date.

B. Approach D: Autonomous Teams

This category is where individual teams are allowed to define their own ways-of-working to facilitate speed, short iterations and delivery quality. This can be seen in domains where by necessity the full product development project cannot move as fast as the individual development teams. Three cases were found that clearly fall into this category [20], [34].

Business: The organisation as a whole is focused on physical delivery of a product (or thousands of products) to the customer, so the ability of individual teams to continuously deploy new software is not seen in the business. Project risk management is focused on minimising technological risks.

Architecture: The architecture of the product is tailored towards satisfying product qualities, requiring a lot of architecting effort spent on defining, verifying and maintaining interfaces, as well as integration of subsystems throughout the project.

Process: Since the overall R&D approach used in the organisation is governed by a stage gate process, or V-model, effort is spent on aligning the practices of the individual teams to the overall process, as described by [20]. The short iterations on the module level are never visible in the large stage-gate process, since the deliveries are still planned towards

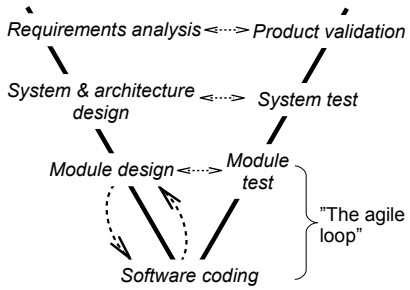

Figure 2. The fast-looping iterations of individual teams are isolated to a small part of the overall V-process of the large project.

scheduled integration points. If the teams adopt e.g. any agile software practices these are only seen at a module or sub-system level, where the short iterations take place, as described in Figure 2.

Organisation: The initiative for the teams to be more self-directed in terms of development practices usually comes from the teams themselves, e.g. they want to adopt agile practices from XP or Scrum [20]. This means these teams are more or less isolated in their approach compared to other teams they are interacting with.

1) Example: Climate Control Software: The case concerns in-house development of climate control software at VCC, where it was outsourced for the previous generation cars. The target car is not in production at the time of publication. The study was performed to provide feedback to software process improvements at VCC, and is previously published in [7].

It was the team themselvs that wanted to use agile practices, so the initiative came from the "grass-roots" of the organisation. The upper management of R&D approved this for the following reasons:

- Shorter lead-times: Having the ability to introduce new features in a controlled manner with the right quality close to launch date of new year models.
- Better to have 80% of the desired software with really good quality than 100% so-and-so.
- Increase in competence: Focus on development teams with continuous learning and improvement will increase the level of competence among the employees.
- Attractive workplace: There are reports of other organisations introducing agile methods having more satisfied employees. Software engineers learn these methods at universities today, it will be easier to recruit and retain competent people.
- A natural progression of the software process improvements already being implemented.

The software runs on a hardware platform with basic software delivered hardware supplier of the HVAC (Heating, Ventilating, and Air Conditioning) Most of the algorithms are developed in Simulink from which C code is generated. Both the control software and the standardised basic software are based on the AUTOSAR software architecture [35] and the interfaces to other systems, including HMI, are stable.

The development team applies most of Scrum practices [36] [37] since this was a natural evolution of present team practices, especially in the light of going from outsourcing to in-house development. The team adjusted their sprint schedule to suit the integration events of the complete electrical system. An unforeseen benefit seen already after three sprints was better prediction of future gate fulfilment compared to previous ways-of-working.

The governance structure was simple with few different concerned stakeholders. The product owner resides at the interior department, in cooperation with one person from product planning at VCC. The development team of nine persons and the Scrum master are part of the electrical department with extensive domain expertise of climate control.

This case is a typical description of an isolated team trying to achieve shorter feedback loops within a large industrial project, i.e. using a D approach. The team defined their way-of-working, while still meeting the overall stage-gate process. Contributing factors to the success was a stable architecture, especially in the interfaces to other sub-systems, and good domain knowledge among the developers.

C. Approach C: Adaptive Processes

At this position an organisation adapts its overall product development process to utilise the possibilities software offers compared to hardware development and manufacturing. This does not seem to be a common position for an organisation to operate at, none of the found cases mention e.g. continuous deployment from autonomous teams, but one case describes how software and hardware development "work according to different methodologies (which) makes it harder to synchronize the work between them." [38].

Business: The development is focused on technology and technological innovation while still preserving key product quality attributes common in the domain.

Architecture: Similar to the previously described positions; the architecture of the product is tailored towards satisfying product qualities.

Process: The process adaptation can take many forms, but a typical measure is to adjust the schedule to the size and scope of what is being developed, and as a result the software and hardware development processes are usually decoupled. This in turn enables software deployment independent of the hardware manufacturing, even to the point of deployment post-manufacturing.

Organisation: Even more so compared to the previous approach the teams are self-directed, both in their ways-of-working and also in defining the deployment date for their deliverables.

D. Approach B: Architecture for Composition

This approach is not very common either, only two cases falls into this category [11], [12].

Business: The business is focused on minimising risk with technology investments, i.e. the R&D organisation has adapted its architecture, processes and organisation to fast development in short iterations, but the product management and business model is still "traditional", focusing on delivering and getting paid for each product.

Architecture: The main difference compared to the previous approaches is a shift in what qualities are emphasized when designing the architecture. In previous approaches there is an emphasis on product qualities discernible at run-time, including cost of ownership, but at this approach qualities affecting the speed, effort and cost of the development are weighted against the former two categories. Typical quality attributes driving the architecture are composability, deployability, maintainability and configurability. These are not immediately discernible for the end user, but facilitate desired ways of working for the development organisation and their teams. There is still value in the product-centric quality attributes, and some may still be vital in heavily regulated domains, but organisations balance these against the development-centric attributes. Typically the architecture style is application- or component-based on top of a supporting software platform providing both infrastructure mechanisms and domain-specific services utilised in new innovations. The software platform provides hardware abstractions of sensors and actuators and also executes control over which features can access various hardware. The platform evolves on a different, usually slower, schedule compared to the products utilising it, and may be supplied by a separate organisation than those developing products.

Process: The process is similar to the previous approach, with the organisation as a whole develop software in short iterations and likely applying continuous integration of software.

Organisation: Similar to the previous approach teams are self-directed.

E. Approach A: Marlin Organisations

There is just one case that describes an organisation working with this R&D approach [12]. This case is borderline of what can be called an embedded system, describing mobile smart phones. Nevertheless this category is important as it may be a forerunner to more "open" embedded products utilising a development approach through an open software ecosystem.

Business: One of the major business drivers in this category is to minimise the risk associated with offering the wrong product or developing undesirable features. Software is seen as a major differentiator in the ability to attract and maintain customers, and therefore short leadtimes from idea to deployment is needed to stay competitive.

Architecture: The architectures used optimise the ability to develop products and features with the shortest possible leadtime. Typical properties of the architecture are to emphasise modularity and composability of software from different teams. Standardised platforms are the norm, either industry-wide encompassing 3rd parties or developed within the large organisation.

Process: The process model used is highly iterative aiming to take small development steps and validate these in each evolution, in some cases with real customer feedback. Activities such as requirements elicitation, design, implementation and verification & validation are done within each short iteration. Planning follows a template based on continuous evolution of the product instead of calendar time. For software development a wide-spread agile process such as XP or Scrum can be used.

Organisation: The organisation consists of self-directed and self-organised teams containing cross-functional competences of product management, architecture, design & implementation and testing, capable of invention and launch of new features. The teams operate autonomously and coordination and integration of interfaces is ensured by the underlying platform and its architecture, allowing the teams to choose their own process and activities. Development of new features realised by software is not outsourced since this is seen as a highly competitive skill.

V. Related Work

[26] presents a systematic literature review of alignment between the four dimension of BAPO, and conclude there is a research gap on alignment of BAPO software product development. The short paper does not prescribe any model for organisations to aid alignment or adjust misalignment. The model in Figure 1 described how companies have aligned their BAPO concerns in the framework of Betz and Wohlin.

[39] presents the "stairway to heaven", a pattern over how companies evolve their software development practices. A comparison by their five steps with the model in Figure 1 shows almost a 1-to-1 mapping, with "Traditional development" corresponding to E in the model, "Agile R&D organization" to D, "Continuous integration" to C, and "Continuous deployment" and ´´R&D as an experiment system" to A. The two models complement each other since they have a different focus: The "stairway to heaven" describes how the software development iterations shortens and involve customer collaboration, while the model presented here describes how the process dimension relates to the other dimensions of business, architecture and organisation.

There are frameworks to analyse software processes such as the Zachman framework [40] and others based on this, for example [41] and [42]. The former of these two is similar to what is proposed here, but has a narrower scope on just software processes. The latter does not define concrete approaches and therefore does not support identification of "where to be". Both frameworks provide some dimensions used to analyse an organisation, but does not propose any possible positions or movements along those dimensions.

[43] use a framework approach to analyse and optimise IT-systems and business processes for mobile business-to-employee-applications for large workforces. They give both some example cases and provide some typical usages of the framework, e.g. understanding business objectives, calculating return on investment, analyse requirements, and model and optimise processes.

The characteristics used to distinguish between "Traditional approaches" and "Internet/intranet development" in [44] spans the same range of software development approaches as Figure 1. The two approaches could be interpreted as

a simplification of the five positions proposed above. The mitigation strategies of the deadly risks would then suggestions of how to evolve along the spectrum of positions.

[45] explores the context of agile development, providing an analysis framework, but has no explicit prescription on how to introduce agile development outside what he calls "the agile sweet-spot". In essence he explores in depth approach D above and how suitable it is for various types of development. [46] identifies a set of six limitations with agile development. They are of concern when developing software with approach D above. [47] describes how to scale agile practices, specifically program management and product backlog administration, in large organisations with several development teams contributing to the final product. This would correspond to evolving from approach D to C in the model above.

VI. Conclusions

Fast development is a competitive advantage also for MPES products. But when the manufacturing-driven development process also is applied to software development it may cause outdated functionality at time of product introduction. Several studies report on successful implementation of agile methods in development of software systems with strong user interaction, e.g. web-based shops [48], [49]. However, in the domain of MPES iterative development aiming at minimising the risk of delivering unused or unwanted user features is the uncommon exception. In this paper, we provided an overview of the problem context of software development of MPES, with distinguishing factors such as the co-design of software and hardware, strong focus on manufacturing aspects, supplier involvement and safety-critical functionality. The complexity of the projects in this domain suggests a need of a holistic model to choose a suitable approach of software development. We performed a mapping study to identify industrial software development approaches, and could identify a model consisting of five distinct approaches. They ranged from "traditional" stage-gate projects focusing on product qualities and large integration efforts, to fast development in short iterations by autonomous teams based on a composable software platform.

The first key contribution of the paper is the model of five archetypical approaches of embedded software development, all which have been used in industry. The model describes how the concerns of business, architecture, process and organisation were aligned in found projects. The model was elucidated by two empirical cases from Volvo Car Corporation. The second contribution is the empirically grounded insight that successful software projects have aligned their business, architecture, software processes and organisation, while misalignment seems to cause difficulties even though each dimension in isolation seems reasonable. A first-hand case was presented were the dimensions were not aligned, which caused difficulties in the project. The third contribution is the fact the mapping study suggests a direction of how organisations evolve from one approach to another. The evolution is driven both by extrinsic factors, such as changed business and innovation goals, and by intrinsic factors such as the inherent complexity in managing and integrating the work of an increasing number of development teams.

Acknowledgment

This work has been financially supported by the Swedish Agency for Innovation Systems (VINNOVA) and Volvo Car Corporation within the partnership for Strategic Vehicle Research and Innovation (FFI).

References

[1] C. Ebert and C. Jones, "Embedded software: Facts, figures, and future," *Computer*, vol. 42, no. 4, pp. 42–52, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5054871&isnumber=5054856

[2] P. Liggesmeyer and M. Trapp, "Trends in embedded software engineering," *IEEE Software*, vol. 26, no. 3, pp. 19–25, 2009.

[3] P. Manhart and K. Schneider, "Breaking the ice for agile development of embedded software: An industry experience report," in *Proceedings of International Conference on Software Engineering*. Washington, DC, USA: IEEE, 2004, p. 378–386. [Online]. Available: http://dl.acm.org/citation.cfm?id=998675.999442

[4] J. Bosch and U. Eklund, "Eternal embedded software: Towards innovation experiment systems," in *Proceedings of the International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, ser. Lecture Notes in Computer Science, vol. 7609. Heraclion, Crete: Springer, 2012, p. 19–31. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34026-0_3

[5] U. Eklund and C. M. Olsson, "A case study of the architecture business cycle for an in-vehicle software architecture," in *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*. Cambridge, UK: IEEE, 2009, pp. 93–100. [Online]. Available: dx.doi.org/10.1109/WICSA.2009.5290795

[6] U. Eklund and H. Gustavsson, "Architecting automotive product lines: Industrial practice," *Science of Computer Programming*, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2012.06.008

[7] U. Eklund and J. Bosch, "Applying agile development in mass-produced embedded systems," in *Agile Processes in Software Engineering and Extreme Programming*, ser. Lecture Notes in Business Information Processing, vol. 111. Malmö, Sweden: Springer, 2012, pp. 31–46. [Online]. Available: http://www.springerlink.com/content/n90815g481713091/

[8] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, and H. Obbink, "Software product family evaluation," in *Proceedings of the Software Product Line Conference*, ser. Lecture Notes in Computer Science, vol. 3154. Springer, 2004, pp. 110–129. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-28630-1_7

[9] T. Brannick and D. Coghlan, "In defense of being "Native", the case for insider academic research," *Organizational Research Methods*, vol. 10, no. 1, pp. 59–74, 2007. [Online]. Available: http://orm.sagepub.com/cgi/content/abstract/10/1/59

[10] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and University of Durham, Tech. Rep. EBSE-2007-01, 2007. [Online]. Available: http://www.dur.ac.uk/ebse/guidelines.php

[11] U. Eklund and J. Bosch, "Introducing software ecosystems for mass-produced embedded systems," in *Proceedings of the International Conference on Software Business*, ser. Lecture Notes in Business Information Processing. Cambridge, MA, USA: Springer, 2012, pp. 248–254. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-30746-1_20

[12] H. Hartmann, T. Trew, and J. Bosch, "The changing industry structure of software development for consumer electronics and its consequences for software architectures," *Journal of Systems and Software*, vol. 85, no. 1, pp. 178–192, 2012.

[13] R. A. McGee, U. Eklund, and M. Lundin, "Stakeholder identification and quality attribute prioritization for a global vehicle control system," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. Copenhagen, Denmark: ACM, 2010, pp. 43–48.

[14] R. K. Yin, *Case Study Research: Design and Methods*, 3rd ed. Sage Publications, 2003.

[15] J. Axelsson, "Towards a process maturity model for evolutionary architecting of embedded system product lines," in *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. ACM, 2010, p. 36–42, ACM ID: 1842764.

[16] N. Mellegård and M. Staron, "Characterizing model usage in embedded software engineering: A case study," in *Proceedings of the European Conference on Software Architecture: Companion Volume*. Copenhagen, Denmark: ACM, 2010, pp. 245–252. [Online]. Available: http://dl.acm.org/citation.cfm?id=1842800

[17] J. Axelsson, "Evolutionary architecting of embedded automotive product lines: An industrial case study," in *Proceedings of the Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture*. Cambridge, UK: IEEE, sep 2009.

[18] D. Sundmark, K. Petersen, and S. Larsson, "An exploratory case study of testing in an automotive electrical system release process," in *Proceedings of the IEEE International Symposium on Industrial Embedded Systems*, Västerås, Sweden, 2011, pp. 166–175.

[19] G. Selim, S. Wang, J. Cordy, and J. Dingel, "Model transformations for migrating legacy models: An industrial case study," in *Proceedings of the European Conference on Modelling Foundations and Applications*, vol. 7349. Lyngby, Denmark: Springer, 2012, pp. 90–101. [Online]. Available: http://www.springerlink.com/content/d136557677552666/

[20] D. Karlström and P. Runeson, "Integrating agile software development into stage-gate managed product development," *Empirical Software Engineering*, vol. 11, no. 2, pp. 203–225, 2006. [Online]. Available: http://www.springerlink.com/content/1r77g17h388472nm/abstract/

[21] S. Müller, P. Kræmmergaard, and L. Mathiassen, "Managing cultural variation in software process improvement: A comparison of methods for subculture assessment," *IEEE Transactions on Engineering Management*, vol. 56, no. 4, pp. 584–599, 2009.

[22] F. Rosa, R. Miniscalco, F. Rame, and F. Torchia, "An on-board software approach to manage a complex space mission and its FDIR: the herschel-planck case study." Edinburgh, Scotland: European Space Agency, 2005, pp. 341–344.

[23] H. Gustavsson and J. Axelsson, "Architecting complex embedded systems: An industrial case study," in *Proceedings of the IEEE International Systems Conference*. IEEE, 2011, pp. 472–478. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5929054&isnumber=5929032

[24] S. Pantsar-Syväniemi, J. Taramaa, and E. Niemelä, "Organizational evolution of digital signal processing software development," *Journal of Software Maintenance and Evolution*, vol. 18, no. 4, pp. 293–305, 2006.

[25] H. Hartmann, M. Keren, A. Matsinger, J. Rubin, T. Trew, and T. Yatzkar-Haham, "Using MDA for integration of heterogeneous components in software supply chains," *Science of Computer Programming*, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2012.04.004

[26] S. Betz and C. Wohlin, "Alignment of business, architecture, process, and organisation in a software development context," in *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 2012, p. 239–242. [Online]. Available: http://doi.acm.org/10.1145/2372251.2372295

[27] K. Kim, H. Kim, and W. Kim, "Building software product line from the legacy systems "experience in the digital audio & video domain"," in *Proceedings of the International Software Product Line Conference*. IEEE, 2007, pp. 171–180. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4339266&isnumber=4339240

[28] E. Johansson, J. Nedstam, F. Wartenberg, and M. Höst, "A qualitative methodology for tailoring SPE activities in embedded platform development," in *Proceedings of the Conference on Product-Focused Software Development and Process Improvement*, ser. Lecture Notes in Computer Science, vol. 3547. Oulu, Finland: Springer, 2005, pp. 39–53. [Online]. Available: http://www.springerlink.com/content/t9qa6qx2ue78dh2k/

[29] P. Wallin, S. Johnsson, and J. Axelsson, "Issues related to development of E/E product line architectures in heavy vehicles," in *Proceedings of the Annual Hawaii International Conference on System Sciences, HICSS*. Big Island, HI, USA: IEEE, 2009, pp. 1–10.

[30] A.-E. Rugina, D. Thomas, X. Olive, and G. Veran, "Gene-auto: Automatic software code generation for real-time embedded systems," in *Proceedings of DAta Systems In Aerospace Conference*. European Space Agency, 2008.

[31] A. Winkler, L. Günther, G. Machel, and M. Schade, "Reuse of the core EPM EGSE in the ground segment," in *Proceedings of Data Systems in Aerospace Conference*. Edinburgh, Scotland: European Space Agency, 2005, pp. 465–468.

[32] R. Cooper, "Stage-gate systems: A new tool for managing new products," *Business Horizons*, vol. 33, no. 3, pp. 44–54, 1990.

[33] M. Cooperation, "Media oriented systems transport (MOST)," 2008. [Online]. Available: http://www.mostcooperation.com/

[34] F. Mafakheri, F. Nasiri, and M. Mousavi, "Project agility assessment: An integrated decision analysis approach," *Production Planning and Control*, vol. 19, no. 6, pp. 567–576, 2008.

[35] S. Fürst, J. Mössinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkämper, G. Kinkelin, K. Nishikawa, and K. Lange, "AUTOSAR - a worldwide standard is on the road," in *International VDI Congress Electronic Systems for Vehicles*, Baden-Baden, Germany, 2009. [Online]. Available: http://www.autosar.de/download/papersandpresentations/AUTOSAR%20-%20A%20Worldwide%20Standard%20is%20on%20the%20Road.pdf

[36] K. Schwaber, "Scrum development process," in *Proceedings of the ACM Conference on Object Oriented Programming Systems, Languages, and Applications*, 1995, pp. 117–134.

[37] H. Kniberg, *Scrum and XP from the Trenches*. C4Media, 2007. [Online]. Available: http://www.crisp.se/bocker-och-produkter/scrum-and-xp-from-the-trenches

[38] A. Shatil, O. Hazzan, and Y. Dubinsky, "Agility in a large-scale system engineering project: A case-study of an advanced communication system project," in *Proceedings of the IEEE International Conference on Software Science, Technology, and Engineering*. IEEE, 2010, pp. 47–54.

[39] H. Holmström Olsson, H. Alahyari, and J. Bosch, "Climbing the "Stairway to heaven"," in *Proceeding of the Euromicro Conference on Software Engineering and Advanced Applications*, Cesme, Izmir, Turkey, 2012.

[40] J. A. Zachman, "A framework for information systems architecture," *IBM Systems Journal*, vol. 26, no. 3, pp. 276–292, 1987.

[41] J. Lonchamp, "A structured conceptual and terminological framework for software process engineering," in *Proceedings of the International Conference on the Software Process*, Berlin, Germany, 1993, pp. 41–53.

[42] J. Feller and B. Fitzgerald, "A framework analysis of the open source software development paradigm," in *Proceedings of the International Conference on Information Systems*, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=359723

[43] V. Gruhn and A. Köhler, "Analysing and enhancing business processes and IT-systems for mobile workforce automation: a framework approach," in *Proceedings of the 2007 Euro American conference on Telematics and information systems*. ACM, 2007, pp. 26:1–26:8. [Online]. Available: http://doi.acm.org/10.1145/1352694.1352721

[44] D. Reifer, "Ten deadly risks in internet and intranet software development," *IEEE Software*, vol. 19, no. 2, pp. 12–14, 2002.

[45] P. Kruchten, "Contextualizing agile software development," *Journal of Software: Evolution and Process*, 2011. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/smr.572/abstract

[46] D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," *Systems Engineering*, vol. 43, pp. 43–46, 2002.

[47] M. Laanti, "Implementing program model with agile principles in a large software development organization," in *Proceedings of the International Conference on Computer Software and Applications*. Turku, Finland: IEEE, 2008, pp. 1383–1391. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4591786&tag=1

[48] D. Goodman and M. Elbaz, ""It's not the pants, it's the people in the pants" learnings from the gap agile transformation - what worked, how we did it, and what still puzzles us," in *Proceedings of the Agile Conference*. IEEE, 2008, pp. 112–115.

[49] M.-W. Chung and B. Drummond, "Agile at yahoo! from the trenches," in *Proceedings of the Agile Conference*. IEEE, 2009, pp. 113–118.