How About Agile Systems Development?

Hermann Kaindl, Edin Arnautovic, and Jürgen Falb
Institute of Computer Technology
Vienna University of Technology
Vienna, Austria
{kaindl, arnautovic, falb}@ict.tuwien.ac.at

***Abstract*—In recent years, a hype about "agile" *software* development has been growing. So, there may be some temptation to simply transfer such approaches to general systems development, for system including *hardware*. It is important to understand, however, that a core idea behind agile software approaches is *iterative and incremental development* (IID), actually an old and proven idea. Unfortunately, increments especially in rapid iterations face inherent limitations in hardware development. So, we claim that agile development for general systems involving hardware is much more difficult to achieve than what current folklore may assume. In order to address this issue, we constructively propose a development life cycle for general systems that takes these limitations into account. It distinguishes between iterations with major, minor and with almost no increments, in order to include hardware development realistically in IID. In this way, some of the promises of agile approaches may be kept in general systems development, whether it is called "agile" or not.**

***Keywords*-Agile development; iterative and incremental development; development life cycle**

I. Introduction

It is important to distinguish between *Agile Systems* engineering and agile *Systems Engineering* [1]. The former deals with *systems* that are flexible, reconfigurable, extensible, scalable, etc. The latter focuses on flexibility and speed in the *process* of conceiving, designing and implementing systems. It is about the ability of the process to respond to new requirements and information during system development.

In this latter spirit, actually *software* development approaches have become popular recently around the notion of "agility", where the most popular today is *Scrum* [2]. A major core idea of these approaches is *iterative and incremental development* (IID), which is actually an old and proven idea.

So, how about "agile" *systems development*? Is it possible to transfer such approaches directly from software to general systems development? Can hardware involved in such systems be developed incrementally in the same rapid iterations as software?

We try to answer these questions in this paper in the following manner. First, we sketch related work on this subject. Then we argue about inherent difficulties arising with hardware increments, which cause issues in agile systems development. Finally, we propose an IID life cycle that takes these limitations with hardware increments into account.

II. Related work

Research in agile systems engineering is still in its infancy, and most of the work investigates the potential utilization of agile approaches in general, or just emphasizes a need for such approaches.

By performing a series of interviews with industry representatives, Stelzmann [3] investigated under which conditions agile Systems Engineering could be used for the development of systems that have a major hardware portion. His results suggest that manufacturing, prototyping and testing of hardware is expensive and takes a lot of time, and implementing changes is hard. For these reasons, his interviewees did not apply agile development for hardware. However, his results also indicate that there is a strong need for more agility in systems engineering, in particular due to dynamic business environments. The industry representatives also claim that agile approaches would better support innovative products as well as complex systems that require more prototyping.

Development of a software-intensive system contains three important aspects according to [4]: business, system, and software. The business aspect is responsible for the economic and operational characteristics of the system including contracting, funding, operational requirements, and overall system delivery structure. The system aspect is responsible for the overall technical and technical management aspects of the system, and the software aspect deals with the software in the system. These authors state that, while some agile approaches have been introduced and executed for business and software aspects, there is a lack of such approaches for the system aspect. We agree with this observation, and claim that caution is needed in transferring these approaches to the engineering of general systems. In addition, these authors argue for development of an agile systems engineering framework but do not give any concrete details how such a framework would look like.

Haberfellner and de Weck [1] propose "Piecemeal Engineering" for the agile adaptation of existing, modular systems. They propose, e.g., to introduce new modules into already existing systems first, and to use such modules for new systems afterwards. Another approach for more agility proposed in the same paper is "Set-Based-Design". Here, designers should work on a set of design alternatives in parallel, and the final design emerges over a sequence of steps (the "best" solution "wins"). However, they do not give any information about how such steps (iterations) should be organized and do not particularly address the specifics of software versus hardware development.

Taxen and Pettersson [5] present *Integration Driven Development* to combine plan-driven, incremental development with agile methods. They define "deltas", which are the incremental system's changes. This approach consists of rigorous planning, but agile realization of such deltas. They propose that such deltas should be implemented in a short time frame (e.g., a couple of weeks), and that this implementation results always in a new working version of the system. It seems, however, that also these authors focus only on software systems (even though large ones) and do not specify how to deal with hardware.

III. Inherent difficulties with hardware increments

Since in our view inherent difficulties with hardware increments pose a key issue for agile systems development, let us briefly sketch them.

First, it is important to understand the difference between iterations and increments in IID. Iterations mean repetitions in the process, whether they involve increments or not. Increments mean extensions of the (software) system in (small) portions.

Such iterations involving increments are at the core of current agile approaches to software development, in particular relatively short iterations. For instance, Scrum involves rapid iterations (with continuous customer input along the way), no longer than one month and usually more than one week.

However, certain inherent properties of hardware development constrain the development of increments. Such properties are, for example, that mechanical parts usually have to be fully built before they can be used in the system, and cabling, which usually has to be done at once for all planned subsystems. This, in turn, requires a final design and a manufactured frame or cover upfront. Further, these properties require a completed design of most of the components before building them. For example, it is necessary to have a good estimate of the power consumption of all electrical parts to be able to specify the required batteries and, in turn, their dimensions.

In general, hardware cannot be built in the timeframe of, e.g., a Scrum iteration. In contrast to software, evolutionary prototyping is usually not possible, since the flexibility required for hardware prototyping is too expensive for production.

Therefore, it is not really possibly to simply apply a current approach to agile software development for the development of general systems involving hardware.

IV. A life cycle model including hardware increments

For agile systems development, we present a life cycle model for systems engineering as illustrated in Figure 1 (we started our work in the context of a prototypical shopping robot [6]). This life cycle model emphasizes the different character of software and hardware development within systems development and recommends what should happen in a controlled development of general systems containing software and hardware. It is iterative and incremental but takes the differences between increments in software and hardware into account.

Although iterations are not necessarily tied to the increments (e.g., the iterations could only improve or polish existing features and not add something new), we focus only on the iterations that are related to increments and implement additional features.

We distinguish between *planned features* and *new features*. Planned features are typically already agreed between stakeholders and prioritized for implementation. New features typically arise from new stakeholder wishes caused by the feedback from the system's users. They are prioritized together with the remaining planned features.

We also distinguish between three types of iterations by the "grade" of the associated increment: iterations with *major*, *minor*, and *almost no* increments. Iterations with major increments are only possible in software as well as for the complete system when next product releases are planned. Iterations with minor, and almost no increments are typical for hardware development.

Let us now give some more process details about the life cycle model as illustrated in Figure 1. It starts with Elicitation of Stakeholder Wishes as a basis of System Requirements Engineering. Defined requirements serve as input for System Architecting, where the overall system architecture is developed. This activity includes also analysis of possible architectural patterns and variants, design space exploration, architectural constraints, etc. We have studied iterations between System Requirements Engineering and System Architecting before [7].

The System Decomposition activity focuses on the breakdown of the system into subsystems and allocation of the planned functionality to the subsystems. Such subsystems typically contain both software and hardware, and can be more or less complex. Depending on their complexity, they may be further decomposed recursively. Finally, each of these subsystems may be further decomposed into software and hardware, which are separately developed in Software Development and Hardware Development activities. Although the allocation of functionality to software and/or

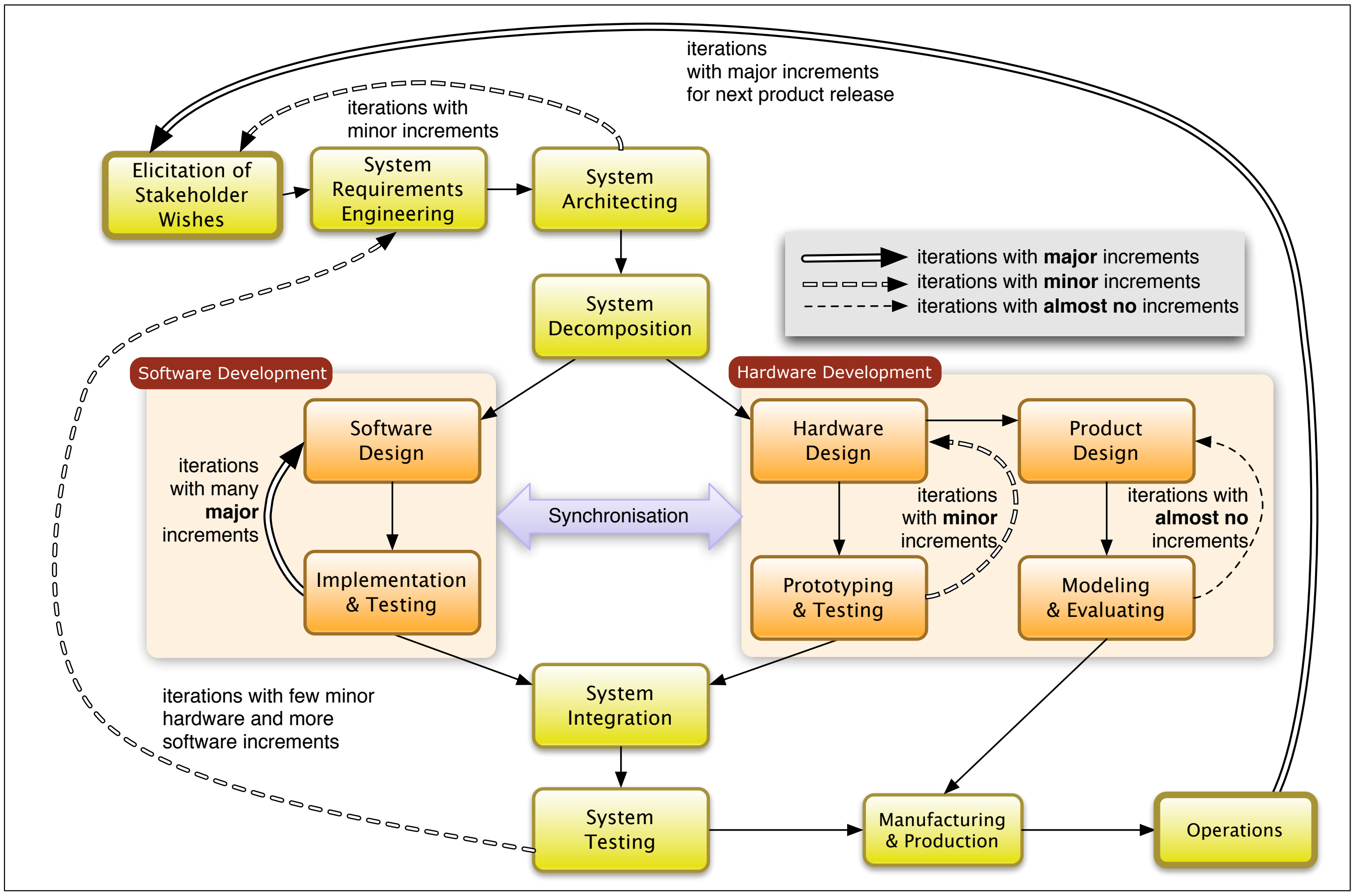
iterations
with major increments
for next product release
iterations with
minor increments
Elicitation of Stakeholder Wishes
System Requirements Engineering
System Architecting
System Decomposition
iterations with major increments
iterations with minor increments
iterations with almost no increments
Software Development
Software Design
iterations with many major increments
Implementation & Testing
Synchronisation
Hardware Development
Hardware Design
Product Design
iterations with minor increments
iterations with almost no increments
Prototyping & Testing
Modeling & Evaluating
System Integration
iterations with few minor hardware and more software increments
System Testing
Manufacturing & Production
Operations

Figure 1. Life Cycle Model for Agile Systems Development

hardware might seem trivial at first (e.g., computations are done in software and movement is performed by drive trains and wheels), this allocation can be challenging. For example, image processing could be done in software but also in (programmable) hardware, and sophisticated techniques could be applied to find a good allocation.

The system architecture and its style, the separation into subsystems and allocation of functionality to hardware or software already define some limitations on increments. When considering software, e.g., the addition of increments in plug-in or component-based architectures is much easier than in a model-view-controller architecture.

The Software Development process might go through many iterations consisting of Software Design and Implementation & Testing activities with *major* increments. The number and size of increments depends mainly on the choice of the software development method. Using a rapid application development process results in more iterations and smaller increments than software development based on the Unified Process. Figure 1 expresses the possibility to include iterations with major increments in the software development process by the double line going from Implementation & Testing back to Software Design. This part of the development may well be a form of evolutionary prototyping. Especially when fast innovation is crucial, evolutionary prototyping can be useful for software development.

Electro-mechanical products are usually more limiting in regard to incremental development. Developing a hardware prototype typically involves the creation of a rather flexible hardware structure based on hardware toolkits, (e.g., mechanic toolkits, or electronic development boards). However, such toolkits are usually too expensive for mass production and do not support appealing product designs. Thus, developing a final product is even more resistive to incremental development. Therefore, we distinguish prototype development from final product development. The two parts in the Hardware Development box in the figure show this distinction.

The first activity, which we call Hardware Design, is dedicated to the development of the mechanical structure of the hardware according to its functionality and the design of the electronic hardware. Such hardware design is both the basis for prototype development built on hardware toolkits and for designing the final product. Prototypes developed here will probably have to be "thrown away", i.e., it will generally not be possible to evolve them to become the final product. Hardware prototyping based on toolkits provides more flexibility and supports multiple iterations with minor increments. It is important to note that the architecture or basic design of the prototype must support increments. It is usually not possible to add major increments without a redesign.

Hardware and software development have to be synchronized in a way that increments added to software and hardware complement each other to allow integration and test of software and hardware components. The so-called anchor point milestones defined for spiral development [8], defining progress of development, may also serve for synchronization of system and software development efforts.

After System Integration, which includes the integration of subsystems into the complete system, there is System Testing. After that, the process can go back to System Requirements Engineering if some of the "planned features" are not yet implemented. Results from product integration and testing can also have other impact on the system requirements, leading to their changes or adding further requirements for the next iteration.

The Product Design activity creates a design of the final product including an appealing look and taking into account materials and guidelines for its manufacturing. Product Design works with producing models and evaluating them with stakeholders and users. Since these models have to give an impression of the final product, iterations have almost no increments. After having tested the functionality with prototypes and evaluated the usability and look based on models (Modeling and Evaluating), the product can be manufactured and put into Operations. For the overall product development life cycle, feedback from the product's users during Operations can lead to new stakeholder wishes and thus to new requirements and major increments for the next product release.

V. Conclusion

This paper argues that "agile" approaches to software development cannot be simply transferred to general Systems Engineering and development of systems involving hardware as well. The reason is that IID cannot be utilized in the same extensive manner as with software, since hardware increments have inherent limitations. Currently, there is a hype of agile software development. Thus it is important to make these inherent limitations clear, so that no naive expectations in "Agile Systems Engineering" may manifest themselves.

This paper also presents a proposal for an iterative and incremental life cycle for general Systems Engineering that takes the limitations of hardware increments into account. Of course, it will yet have to be shown how such an approach can be applied successfully in Systems Engineering practice, and its generality will also have to be investigated yet.

References

[1] R. Haberfellner and O. de Weck, "Agile SYSTEMS ENGINEERING versus AGILE SYSTEM engineering," in *Proceedings of the Fifteenth Annual International Symposium of the International Council On Systems Engineering (INCOSE)*, July 2005, pp. 10–15.

[2] P. Deemer, G. Benefield, C. Larman, and B. Vodde, "The Scrum primer," 2010, version 1.2.

[3] E. Stelzmann, "Contextualizing agile systems engineering," in *Proceedings of the IEEE International Systems Conference (SysCon)*, April 2011, pp. 163–167.

[4] M. R. Kennedy and D. A. Umphress, "An Agile Systems Engineering Process: The Missing Link," *CrossTalk: The Journal of Defense Software Engineering*, vol. 4, no. 3, pp. 16–20, May/June 2011.

[5] L. Taxen and U. Pettersson, "Agile and Incremental Development of Large Systems," in *Proceedings of the 7th European Systems Engineering Conference (EuSEC 2010)*. Stockholm, Sweden: INCOSE, April 2010.

[6] H. Kaindl, J. Falb, E. Arnautovic, and D. Ertl, "Increments in an Iterative Systems Engineering Life Cycle," in *Proceedings of the 7th European Systems Engineering Conference (EuSEC 2010)*, Stockholm, Sweden, April 2010.

[7] H. Kaindl, E. Arnautovic, D. Ertl, and J. Falb, "Iterative requirements engineering and architecting in systems engineering," in *Proceedings of the Fourth International Conference on Systems (ICONS '09)*, March 2009, pp. 216–221.

[8] B. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.