Proceedings of the Hamburg International Conference of Logistics (HICL) – 25

Henning Skirde, Robert Steinert

About the Future Role of Software in the Product

Published in: The Road to a Digitalized Supply Chain Management
Wolfgang Kersten, Thorsten Blecker and Christian M. Ringle (Eds.)
ISBN 9783746765358, September 2018, epubli

About the Future Role of Software in the Product

Henning Skirde[1], Robert Steinert[1]

1 – ID-Consult GmbH

Product development is shifting towards realizing an extended scope of product functions in software. This leads to new challenges in terms of a methodological integration and synchronization of different development disciplines. This paper provides insights on how to systematically manage the harmonization of traditional development disciplines with agile software development based on an integrated data model. To achieve this, the traditional product data model is extended to a seamless system data model that covers – based on a common function structure – the product itself but also services induced by digitalization and infrastructure backends.

Keywords: Product Data Model METUS; Software Development; Agile and SCRUM; Harmonization Digitalization

First recieved: 25.May.2018 *Revised:* 15.Jun.2018 *Accepted:* 04.Jul.2018

1 Software as a Function Owner in Digitalized Products

”Bending sheet metal is not our value creation’s core anymore”, this is – slightly exaggerated – how manufacturing companies put a major change into a nutshell, that has become essential in product development. The functions of technical products are to an increasing extent determined by Software and digital connectivity, “bent sheet metal” does not ensure market success anymore.

Today, software is an indispensable ingredient of technical products. An example is embedded software like in an automobile’s control unit or as a control system of a machine or plant. Nevertheless, traditional development processes based on a two-stage freezing of requirements often lack the consideration of software and do not recognize it as an essential element of the development process according to its role in the completed product.

Mechanics and electronics may feature different development methods, but their common ground and their origin descend from a physical world. Where these disciplines think based on product structures, software development choses a structure based on data models. In addition, software development as a discipline has come up with its own methodology: Agile approaches such as SCRUM (Schwaber & Suther-land 2017) are entirely different from traditional product development processes.

Up to now, a common solution has been to encapsulate the discipline of software development right into the traditional product development process. By this, software development could retain its own approaches and data models. An integration with other development disciplines’ statuses has been conducted based on completed releases, milestones and snapshots. Against the background of major differences regarding methodology and data models of other development disciplines, such an approach seemed to be sufficient. Today on the contrary, such a methodological segregation fails to be a promising way to go (Eigner et al. 2018).

Why is this the case? Software as an ingredient of technical products in the previous sense contains a system related machine control. It might be embedded in a control unit’s firmware with the aim to interpret signals from a machine’s or plant’s sensors and pilots corresponding actuators based on algorithms.

In an easy example, the software of a car’s engine control unit uses the values from sensors at the crankshaft, throttle valve and intake air temperature to calculate

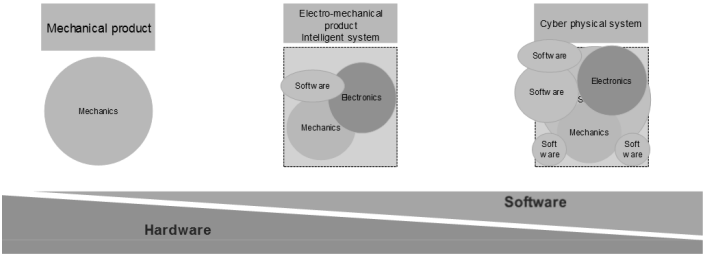

Figure 1: In future products, the software's significance will be enhanced (Source: own representation)

an optimal timing and duration for the fuel injection and subsequently provides the resulting values to the corresponding actors.

This previous sense of software that is despite all complexity well manageable has already been outdated (Figure 1): To this day, software has been modified from an integrated system element to a self-contained function owner. Mainly mechanically determined products have been altered into today's mechatronic products. In the future, due to a higher share of software in products as well as of digital connectivity, this will result in "cyberphysical systems" (Eigner et al. 2016).

2 Software's Promise: More Value from Less Effort

A product's competitive edges can be realized in software to an increasing degree, without changing the hardware in the ideal case. An example from the automotive industry would be an entertainment module that comes with an identical hardware (screen, control unit, operating controls) to minimize the number of physical variants. Particular functions, e.g. a hands-free-speakerphone, can be activated by software. To achieve this, the software covers the maximum functionality. By this, variants can be derived by locking and unlocking individual functions. In this context, software offers the attractive promise of an increased customer value without bearing the costs of a higher physical variance (Khaitan & McCalley 2015).

The opportunities of software as a function owner in a product are numerous and attractive:

1. Variance in software can be realized in a more flexible and more cost-effective way compared to hardware.
2. Additional prospects for differentiation can be derived by additional product functions and services provided by software.
3. Existing products can be updated with more recent software to run new functions. This enhances customer satisfaction and extends the hardware's competitiveness in the life cycle.
4. Elongated operating times regarding hardware, that can be updated with software, reduce the TCO and therefore might be a competitive advantage.

However, these opportunities face several risks:

1. Traditional competitive edges regarding products more and more lose their significance.
2. The high innovation frequency in agile software development can only be transferred to other engineering disciplines with several limitations.
3. Customers have a limited willingness to pay for software due to their experience regarding apps that are free of charge or at least cheap.

Software is not only an essential ingredient of a product – be it as a machine control or as a hands-free-speakerphone's software – but also to an increasing degree a "virtual" system element. This refers to software that an end user will experience as a part of the product functionality, but it is not part of the particular product. To provide an example, an app on the end user's own device like a smartphone will serve as a control for his digitally connected coffee machine.

From this point of view, products increasingly lose their physical boundaries, become more and more digitally connected elements of cyberphysical systems (Eigner et al. 2016). Thereby, new value propositions and business models can be identified and subsequently implemented. But before implemented, both the product development process and also product structures have to be adapted to the new requirements.

For this, the following three areas inhibit new challenges:

1. Methodological integration: The development methods and displaying of requirements in software engineering differ considerably from those in the development disciplines for mechanics and electronics.
2. Integration of "virtual" product components: Components that are a product's ingredient from a functional perspective, but are not in the physical product structure's scope (e.g. a smartphone app for machine control) are not considered in conventional development processes (Conti et al. 2012).
3. Synchronizing the diverse product and innovation cycles among different development disciplines: Agile principles in software development often contain iterations that are not sufficiently synchronized with development cycles of the corresponding hardware.

The following section provides insights on how to systematically manage these three areas, in order to utilize a digital product's advantages to a full extent.

3 Requirements: Traditional, Agile or Both?

Product functionality and innovation are increasingly shifted from mechanics to software. In several areas, mechanical designs have achieved high degrees of maturity. Consequently, disruptive innovations become rather unlikely or require disproportionally high development efforts. Therefore, software can serve as an adequate leverage for new and innovative functionalities.

In the utmost case, the hardware's role is downgraded to a software's operating environment. This has the consequence that requirements to hardware are determined by the dedicated software functionalities. If software becomes a functional driver, it will provide the basis for hardware development. Accordingly, hardware in terms of an operating environment for a software has to be designed with the aim to be able to "carry" an ex ante defined number of software cycles with more and more comprehensive functionality.

This requires enhancing the harmonization of different development disciplines based on a common functional level. This means after initially determining a common function structure, the development team has to decide not only how, but also in which discipline a function shall be realized. In addition, realizing

functions in software also calls for a comprehensive management of requirements that incorporates all disciplines.

Agile software development allows for a realization of a product's market-ready versions in short iteration loops that feature additional functions that are successively implemented in further releases. Compared to this, traditional development disciplines have a higher effort to realize a new product release because an incremental approach comparable to software engineering is not feasible (Vogel & Lasch 2016). The reason for this is the extent to which changes in product's hardware impact the operations – e.g. starting with molding design and subsequently logistics and production.

Agile and traditional methods for requirements management and engineering will thus need to coexist even though a convergence can already be noticed. Against this background, the argument can be derived that traditional and agile requirements processes need to be further harmonized and interrelated to display an overall view on the product system. A seamless product data model plays an important role not only in the course of the entire product development process, but also regarding a common understanding in different development disciplines.

This requires a methodological approach that consolidates a market and product view on the one hand as well as the phases of product definition, design and implementation on the other hand into a common data model that subsequently allows for flexible and numerous visualizations. The METUS methodology and the corresponding software have proven to be appropriate for this (METUS 2018).

4 From Product Architecture to System Architecture

Traditional product architecture models start an entire product's description of with requirements and functions, followed by the product structure consisting of individual function owners that can be both, mechanical and electronical components and modules (Göpfert & Tretow 2013). Without adaptions, such an architecture may not be able to incorporate the external or "virtual" scopes of a product into its structure.

Following the goal of a purposeful enhancement of a product's customer value driven by software functionality (or shifting the realization of functions from hardware (HW) to software (SW)), the traditional product architecture model does not suffice anymore. The model has to be enlarged in order to also manage

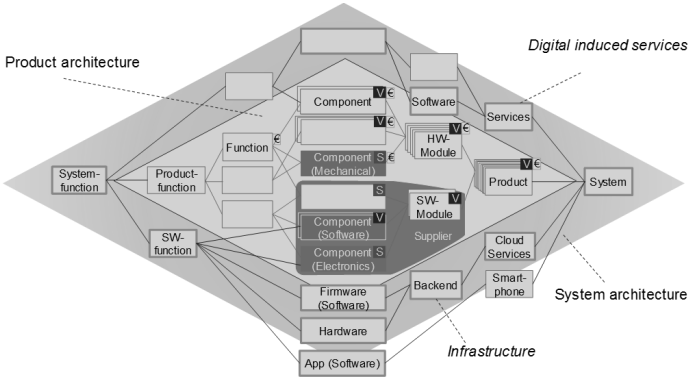

Figure 2: Internal variance is shifted from hardware to software (Source: own representation)

new components and services as an ingredient of the extended architecture. This requires raising the product architecture to a system architecture (figure 2). If this raise is successful, the product's system architecture also covers new "digital" scopes that feature similar characteristics in the function structure like mechanically or electrically realized functions. In addition, this step may also contain changes of the infrastructure, e.g. the provision of a backend or a cloud for value-added services that are required by digitalization (Crawley et al. 2015).

5 Synchronizing Tact Rates of Several Development Disciplines

In a similar way, like software will become a major driver of new product functionalities, the pulsing of software development will increasingly determine the pulsing of the entire development process.

In an age of mainly mechanical products, the product lifecycles corresponded to the innovation cycles (figure 3). Modularizing product architectures has enabled a decoupling of these two different cycles (Skirde 2015; Bahtijarevic et al. 2014; Lau

et al. 2011; Mikkola 2006). Technology and innovation leaps can be realized only in particular modules that replace outdated modules in the original product. The increase of mechatronic products has come with diverging the pulsing of product lifecycle and innovation cycles: New software releases, for example the already explained automobile engine's control unit, are deployed without changing the operating environment provided by the hardware.

To realize an extended scope of product functionalities with software functions, the hardware design has to be adequately dimensioned in advance: it has to be able to operate a software's future releases right from the start. For this, product management has to define up to which future software releases the hardware is supposed to remain unmodified. The ability for updates, known by every end user of a smartphone or tablet computer, will become a feature of nearly every technical product.

The conceptual design of updateable products requires a function roadmap containing software updates planned by the product management. This roadmap has to be transparent for all involved parties and agreed as an ingredient of the entire system architecture represented by a product embedded into this architecture. If this succeeds, new perspectives can be raised: particular scopes of the product variance can be shifted from hardware to software. In total, an extension of the performance spectrum towards a comprehensive product-service-system is likely (figure 4).

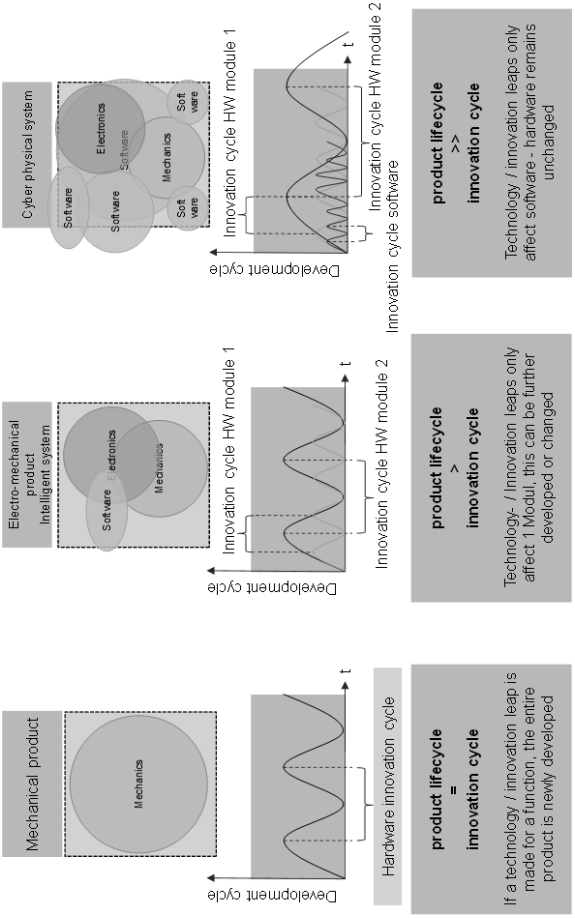

Figure 3: Software's changed significance has an impact on future product development and the entire product lifecycle management (Source: own representation)

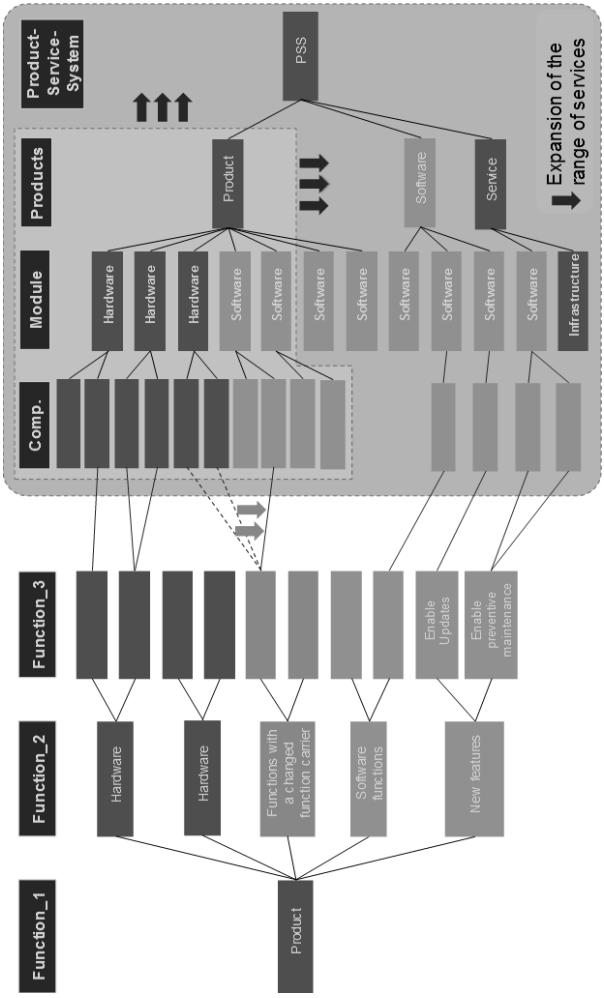

Figure 4: A particular scope of internal variance can be shifted from hardware to software (Source: own representation)

6 Pricing for Software-Based Product Functions

The increasing usage of software as a function owner determines that a higher share of the product requirements has to be fulfilled by software. Consequently, the share of the product's value proposition that has to be covered by software is increased accordingly. From this change, the question can be raised how to achieve an appropriate pricing for software as a function owner.

Today, software development is often accounted into the company's indirect costs. In comparison to the production of physical products, software comes with high non-recurring costs (NRC), but causes only very little recurring costs (RC). In terms of transparency, assigning the costs of individual software development to the corresponding scope of the function structure seems to be more beneficial from an economical perspective than accounting these into the indirect costs (Skirde et al. 2016). An analogous argumentation can be derived for assigning software development costs to requirements as well as the value proposition.

The major success factor that can be derived from a function-based pricing is again the overall view on the product, which is the basis that allows for a cost-oriented analysis of functions:

By connecting the value proposition with requirements (traditional and agile), function structure and finally the product structure with its mechanical, electronical and now also digital scopes, comprehensive data continuity from the market to the product view can be established.

If this procedure in the sense of a function analysis is then conducted the other way around – from the software scope in the direction of the function structure, the software's contribution towards fulfilling a requirement and realizing the value proposition can directly be derived.

The chance to increase a product's customer value by enhancing the software ingredients and thus to realize a greater extent of variance in software on a low level of costs, simultaneously shortening innovation cycles, seems to be exceedingly attractive.

Completely exploiting the chances outlined in this paper will probably only succeed for those companies, that already today turn their attention to a seamless product development process that interconnects a market and a product perspective on a reliable methodological basement.

7 Summary and Conclusion

Product development is shifting towards realizing an extended scope of product functions in software, challenging the past development paradigms. In this paper, the new challenges in terms of a methodological integration and synchronization of different development disciplines have been pointed out. To conclude, our recommendation to companies is to systematically manage the harmonization of traditional and software related development disciplines based on an integrated product data model. A common functional structure representing the entire product can support this harmonization. In addition, it allows for deriving data for the pricing of software related functions and product features. An extension of the product data model can enable a company to consider new services induced by digitalization as well as accordingly required infrastructure scopes in their value proposition.

References

Bahtijarevic, J. and M. De Murcia E Paes (2014). "Applying Modular Function Deployment (MFD) to software architecture". In: *KTH Production Engineering and Management*.

Conti, M., S. Das, C. Bisdikian, M. Kumar, L. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, and F. Zambonelli (2012). "Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence". In: *Pervasive Mobile Computing, Vol. 8 (1)*, pp. 2–21.

Crawley, E., B. Cameron, and D. Selva (2015). "System Architecture: Strategy and Product Development for Complex Sys-tems". In: *Prentice Hall Press, Upper Saddle River, NJ, USA*.

Eigner, M., H. Apostolov, and P. Schäfer (2018). „Nachhaltigkeit 4.0" – Nachhaltige Produktentwicklung im Zeitalter der vier-ten industriellen (R)evolution". In: *Fort-schritte in der Nachhaltigkeitsforschung*, pp. 253–272.

Eigner, M., U. August, and M. Schmich (2016). "Smarte Produkte erfordern ein Umdenken bei Produktstrukturen und Pro-zessen". In: *White Paper, Siemens PLM Software.*

Göpfert, J. and G. Tretow (2013). "Produktarchitektur". In: *In: Feldhusen, J. and & Grote, K.-H. [Eds.]. Pahl/Beitz Konstruktionslehre, 8. Edi-tion, Springer, Berlin/Heidelberg*, p. 252.

Khaitan, S. and J. McCalley (2014). "Design Techniques and Applications of Cyberphysical Systems: A Survey". In: *In: IEEE Systems Journal, Vol. 9 (2)*, pp. 350–365.

Lau, A., R. Yam, and E. Tang (2011). "The Impact of Product Modularity on New Product Performance: Mediation by Product Innovativeness". In: *In: Journal of Product Innovation Management, Vol. 28 (2)*, pp. 270–274.

METUS (2018). "Die METUS Software ist die Schaltzentrale für Produktentscheidungen". In: *[online]: https://www.id-consult.com/loesungen/metus-software.html, called on May 15, 2018*.

Mikkola, J. (2006). "Capturing the Degree of Modularity Embedded in Product Architectures". In: *Journal of Product Innovation Management, Vol. 23*, pp. 128–146.

Schwaber, K. and J. Sutherland (2017). "The SCRUM Guide". In: *[online]: http://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, called January 22nd, 2018.*

Skirde, H. (2015). "Kostenorientierte Bewertung modularer Produktarchitekturen". In: *Eul Verlag, Lohmar-Köln*.

Skirde, H., W. Kersten, and M. Schröder (2016). "Measuring the Cost Effects of Modular Product Architectures — A Conceptual Approach". In: *International Journal of Innovation and Technology Management, 13(4).*

Skirde, H. and R. Steinert (2018). "The Future Role of Software in the Product – Realizing Variance in Software". In: *Whitepaper, DOI: 10.13140/RG.2.2.25525.17126*.

Vogel, W. and R. Lasch (2016). "Complexity drivers in manufacturing companies: a literature review". In: *Logistics Research, 9: 25. doi:10.1007/s12159-016-0152-9*.