

# Lean Innovation to Improve Embedded Software Design and Development

Hans R. Tanner

The continuing integration and miniaturization in microcontroller design along with improved computing power allows for an ever increasing number of new and innovative product functions to be implemented in embedded software. Because of these changes, manufacturers need to prepare themselves to meet new and more complex customer requirements. Unfortunately, the development of embedded software is rarely treated with the same importance as the development of the other product components. Quite often available potentials of modularity, portability, state of the art design techniques, and release management are not exploited to the full extent possible. The techniques introduced by the Lean Innovation methodology can help to achieve significantly better results in this area.

## Embedded Software Development Is often Treated with Lower Priority

Good product design must follow some generally accepted principles: It should start with an in-depth knowledge of the relevant customer requirements, implement a modular design that allows for reuse of modules, meet all applicable regulatory requirements, and comply with applicable technical standards and state of the art practices. The product should be configurable to allow for user configuration or mass customization and meet principles of complexity management, i.e. minimization of the diversity of used parts and variants in both, products and production and logistics processes. Finally, each new product design should be the well thought out result of a higher level, strategy based system architecture and product roadmap.

While significant improvements have been achieved over the last decade in implementing these principles in the tangible parts of products, embedded software development is clearly lagging behind in many cases. Many companies' development processes show the following weaknesses in particular:

- Too often, the development of firmware is treated as an individual project for each device. Opportunities to use commonalities on the level of hardware standardization as well as design tool standardization are not exploited.

### Examples of devices using embedded software:

- A large and constantly growing number of microcontrollers in cars cover all functions.
- 'Ordinary' products like washers, dryers and vacuum cleaners rely on microcontrollers to not only control the user interface, but also the core functions of the product.
- Production equipment like machining centers or injection molders come with built-in web servers that allow for control of the equipment functionality, as well as the analysis of performance data.
- Devices that used to be 'stand-alone' e.g. medical diagnostics devices now have a communication infrastructure that allow to visualize the data on the user's PC and to share it automatically with the doctor.

This list could go on indefinitely, as new devices with improved functionality become available daily.

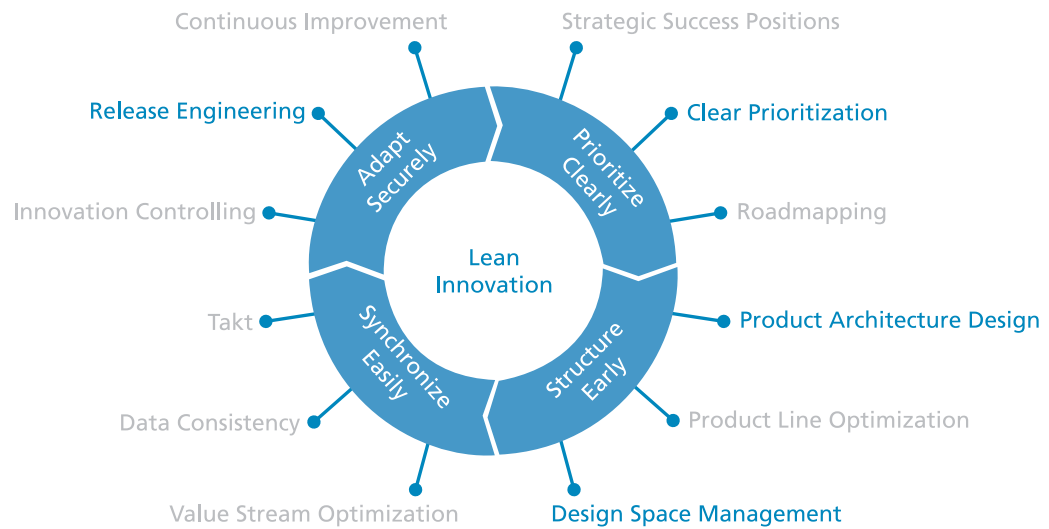


Figure 1: By Simply Focusing on 4 of the 12 Lean Innovation Principles, Embedded Software Design and Development Can Be Improved Significantly

- Modularization is in the best case understood as the availability of function libraries that can be reused. However, an understanding of modularity in the sense of grouping the code along the product functions, even in combination with the related hardware using a proper hardware abstraction and then reusing it in the next product, is rarely done.
- System architectures and product roadmaps rarely treat the development and evolution of low-level software as an individual topic. Technology steps, new technologies and their implementation are seldom identified along a timeline. This is an important topic during the planning stage as there are still frequent technology changes in IT, requiring thorough market observation and technology succession planning.
- Consequently, there is also no harmonization of technologies between a product roadmap and software roadmap. This topic is of particular importance because both worlds typically have very different product life cycles.

Lean Innovation as introduced in the current and past issues of the Complexity Management Journal provides several mechanisms that help to better integrate the development of embedded software with the development of the overall product; a first step to overcome the

above-mentioned problems. In particular, addressing the Value System and Target Hierarchy, Technology Management and Roadmapping, Design Space Management, Product Architecture Design, and Release Engineering can reveal the biggest improvements.

#### Value System and Target Hierarchy

Normally in an engineering driven development environment, not enough time is spent on carefully identifying all aspects of user requirements. This is particularly true regarding embedded software. Even though smart software functionality, e.g. standardized cross-device user interface and menu systems or functions that increase the ergonomics of a device, can greatly contribute to the overall value of the device to a customer. However, the methods that are generally used to recognize customer requirements, such as focus groups, do not help to clearly identify the true requirements. Focus group members think linear and along the options of the sample device. Most likely, their input provides insights into current deficits of a product and ideas for incremental improvements, but rarely will it generate break-through ideas.

As Lean Innovation emphasizes the need to carefully define a clear target hierarchy, additional methods such as the definition of Use Cases, the identification of com-

peting Use Cases, thorough QFD analysis and prioritization should be introduced to break down walls in the design process of embedded software as well.

Furthermore, Lean Innovation promotes the provision of roadmaps as a planning tool for several generations of products, starting with basic technology roadmaps and module roadmaps and ending with product roadmaps. Embedded software should be part of the module roadmap to show the availability of individual versions with specific product features along the product life cycle.

### Design Space Management

In every product development process, there is a significant risk of ruling out potential solutions too early in the process. In many instances, this results of having to reintroduce them again at high costs late in the process. From the embedded software point of view, it is important to keep future expansions alive as long as possible. For example, when implementing a communication interface, it makes absolute sense to use a communication stack that later can be expanded to support additional communication protocols, such as adding wireless communications to a wired device. Furthermore, the design space should also be kept open regarding the attached hardware. This could be achieved by standardizing certain control signals from the outside to the processes that allow for adding new input-based functionality later on in the product life cycle. For example, when implementing a motor drive circuit to set the motor speed of a software or hardware knob, adding a tachometer signal from the motor back to the controller would keep the design space open. In that case, it would be possible later to add a PID feedback loop to create an autonomous control loop in the software without additional hardware changes.

### Product Architecture Design

Just like in mechanical engineering, modularity should be used as a design principle for embedded software as well. Modularity is more than just separating source code in individual units or libraries, e.g. math libraries, but stands for functional separation. Functional modules such as user interfaces, core device functions like control loop implementations or communications protocol

implementation should be kept and treated as individual modules with clearly defined interfaces. They offer the opportunity for improvements and releases on the module level as well as making the hardware-independent fraction of the code portable between devices.

Furthermore, functional modules that have the potential to be reused on additional devices should be separated from the target hardware in order to facilitate the transfer. Examples for such modules are database elements or the structural part of user interfaces and menu systems.

### Release Engineering

The Lean Innovation methodology suggests shortening release cycles to maintain an image of being an innovative solutions provider. Software can facilitate short release cycles to a great degree, as improvements to the product can be implemented at relatively low cost. Therefore, it is important to overcome the 'hotfix' mentality, carefully plan and make use of short release cycles and upgrades made available by embedded software.

When carefully planned, a company can introduce a new device using a standard hardware where not all hardware functions are supported by software at the beginning. By providing software releases, most often installable by the user through Internet-based distribution, functionality can be added to the device over time, thereby increasing the value for the customer.

### Conclusion

In our experience, the design of embedded software is often treated as a topic of minor importance in the design process of typical hardware devices or machinery. As a result, available potentials of smart and elaborate software functionality are not fully exploited. Applying the Lean Innovation principles to the design process of embedded software can help to overcome these weaknesses, turning embedded software from a necessity to a core contributor of product success.

#### Contact

**Stephan Krumm, Ph.D.**

Phone: +49 2405 459 02

[stephan.krumm@schuh-group.com](mailto:stephan.krumm@schuh-group.com)