Expertise
and insight
for the future

Antti Teronen

Improvement of Embedded System Automated System Testing

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

8 May 2019

PREFACE

Special thanks to my wife for all the support and loving care. I spent several weekends and vacation weeks studying and writing. Despite all the skipped family events and missed activities together she still smiles at me.

Also, our dog helped me to get this thesis ready, and I'm grateful to her. She kept watch beside me, wasn't annoyed (too much) about the clicking keyboard noise and reminded me about the importance of lunch breaks and walks in the outdoors.

It's also worth mentioning that this thesis would not have been possible to complete without the constant delays in national railroad services and the local grocery stores home style takeaway food. These gave me the extra time slots and energy to finalize this thesis.

Lahti, 8 May 2019
Antti Teronen

Author Title Number of Pages Date	Antti Teronen Improvement of Embedded System Automated System Testing 52 pages + 3 appendices 8 May 2019
Degree	Master of Engineering
Degree Programme	Information Technology
Professional Major	Health Technology
Instructors	Niko Nuotio, Engineering Manager Sakari Lukkarinen, Senior Lecturer

Patient monitors are complex embedded systems. They integrate analog and digital electronics and several microcontrollers and processors as one system, which is controlled by embedded software and operating system drivers. Clinical application software running on top of the system gathers all measured physiological data, generates waveforms and visualizes everything to the user on the monitor screen.

Medical device development is highly regulated. The responsible organization must demonstrate that the medical device is safe and effective. Company shall have a quality system in place to demonstrate that the development and maintenance process meets the regulatory requirements. Also, verification and validation results are required to show compliance with relevant laws and standards. As part of the product verification, automated system testing can be used to speed up the development process and to improve testing coverage.

System testing is performed for the whole patient monitoring system. Test cases are run several times during product development for various product iterations. In a sophisticated modular system like a patient monitoring system, testing the functionality of each interface and feature makes the testing system also complicated. Ideally, the system test automation should have the capability to make all the same tasks as the end user. These tasks include powering on, shutting down or adding and removing system peripherals. Also, the test automation system should gather as much as possible data about the system software and hardware to find the correlation between system behavior and found defects. All available data helps to find the defect root cause, make corrections and re-test as fast as possible.

This study concentrated on embedded system hardware testing, types of defects observed and their root causes. Purpose of this study was to gather data and search for information to create a list of improvements to consider when adding new functionalities to the automated testing system. New functionalities should be such, that they enhance the test system capabilities to emulate real user interaction and scenarios, and they should help to reveal underlying hardware related defects.

Keywords	embedded system, automated system testing, electronics design process

Tekijä Otsikko Sivumäärä Päiväys	Antti Teronen Improvement of Embedded System Automated System Testing 52 sivua + 3 liitettä 8.5.2019
Tutkinto	Master of Engineering
Koulutusohjelma	Information Technology
Suuntautumisvaihtoehto	Health Technology
Ohjaajat	Niko Nuotio, Engineering Manager Sakari Lukkarinen, Senior Lecturer

Potilasvalvontamonitorit ovat monimutkaisia sulautettuja järjestelmiä. Järjestelmään kuuluu analogista ja digitaalista elektroniikkaa, sekä useita mikrokontrollereita ja prosessoreja, jotka ohjaavat järjestelmän toimintaa sulautetun ohjelmiston ja käyttöjärjestelmän laiteohjainten avulla. Järjestelmän kliininen ohjelmisto kerää mittausdatan potilaan elintoiminnoista, luo siitä käyrät ja näyttää kaiken monitorin näytöllä käyttäjälle.

Lääkintälaitteiden kehitys on vahvasti säänneltyä. Tuotteesta vastuussa olevan valmistajan täytyy kyetä osoittamaan, että laite on turvallinen ja sopiva käyttötarkoitukseensa. Valmistajalla tulee olla käytössä laatujärjestelmä, joka osoittaa että tuotekehitys- ja ylläpitomenetelmät täyttävät viranomaisvaatimukset. Lisäksi lakien ja säännösten vaatimustenmukaisuuden osoittamiseksi tarvitaan todistusaineisto tuotteen testauksesta ja todennuksesta. Tuotteen testauksen osana voidaan käyttää automaattista järjestelmätestausta. Automaattinen testaus nopeuttaa tuotteen kehitystä ja parantaa testauksen kattavuutta.

Järjestelmätestaus suoritetaan koko potilasvalvontamonitorijärjestelmälle. Testitapaukset suoritetaan useaan kertaan tuotekehityksen aikana tuotteen eri kehitysversioille. Koska potilasvalvontajärjestelmä on modulaarinen ja monimutkainen, järjestelmän jokaisen rajapinnan testaaminen johtaa myöskin monimutkaiseen testausjärjestelmään. Ihannetapauksessa testausjärjestelmän pitäisi kyetä käyttämään järjestelmää kuten oikea loppukäyttäjä. Loppukäyttäjä voi esimerkiksi käynnistää ja sammuttaa järjestelmän sekä poistaa tai liittää sen lisälaitteita. Oikeiden käyttötapausten jäljittelyn lisäksi testausjärjestelmän pitäisi kyetä keräämään mahdollisimman paljon tietoa järjestelmän laitteiston ja ohjelmiston toiminnasta. Kerätyn tiedon avulla voidaan selvittää mistä löydetyt järjestelmävirheet johtuivat, korjata viat, sekä päästä testaamaan korjaukset uudestaan mahdollisimman nopeasti.

Tässä työssä keskityttiin sulautettujen järjestelmien laitteiston testaukseen, millaisia virheitä siinä voi esiintyä sekä mistä ne johtuvat. Työn tarkoituksena oli kerätä tietoa automaattisen järjestelmätestauksen laajennusten ja parannusten suunniteluun. Laajennukset ja parannukset tulisi olla sellaisia, että ne mahdollistavat oikeiden käyttötapausten paremman jäljittelyn, sekä parantavat mahdollisuuksia löytää virheitä testattavan järjestelmän laitteistosta.

Avainsanat	sulautettu järjestelmä, automatisoitu järjestelmätestaus, elektroniikan tuotekehitysprosessi

Contents

List of Abbreviations

3D	Three Dimensional
AC	Alternating Current
ACM	Association for Computing Machinery. Worldwide association for educational and scientific computing society for delivering resources for advancing computing as a science and a profession.
AD	Analogue to Digital
API	Application Programming Interface. Set of methods, routines, and protocols to interact with an application.
ASCII	American Standard Code for Information Interchange. Standardized coding for a character set.
ATDD	Acceptance Test Driven Development. Development method where acceptance tests are written in advance of implementing the corresponding functionality.
BGA	Ball Grid Array. Type of integrated circuit package where soldering balls are arranged on the bottom of the package as an array.
CPU	Central Processing Unit
DA	Digital to Analogue
DC	Direct current
GPIB	General Purpose Instrument Bus. Another name for IEEE-488 short-range digital communications 8-bit parallel multi-master interface bus.
GUI	Graphical User Interface
HALT	Highly Accelerated Lifetime Testing. Stress testing method for enhancing product reliability and finding products weak links.
HW	Hardware. Typically in electronics development, the hardware means the printed circuit board assembly and peripherals it is interacting with. In some cases, hardware may mean the whole device electronics and mechanical parts.

IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IEEE-488	Specification for short-range digital communications 8-bit parallel multi-master interface bus.
IID	Iterative and Incremental Development. Method for developing product features in development iterations and adding new ones incrementally.
Java	A class-based and object-oriented, general-purpose software programming language.
LabView	LabView (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming environment from National Instruments Corporation and is based on the concept of data flow programming.
MCU	Micro Controller Unit
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
MySQL	Open source relational database management system. SQL is an acronym of Structured Query Language.
PCB	Printed Circuit Board. Mechanical support and conductive wiring for connecting electronics components as a circuit.
PCI	Peripheral Component Interconnect. Parallel bus type interface used in microprocessors to connect embedded peripherals and add-on cards.
PCIe	Peripheral Component Interconnect Express. Serial bus type interface used in microprocessors to connect embedded peripherals and add-on cards.
PDN	Power Distribution Network
PXIe	PCI eXtensions for Instrumentation (PXI) Express. Common form factor and backplane bus for modular test and measurement equipment. PXIe is an adaptation of PCI Express bus to the PXI mechanical form factor.
Python	An interpreted high-level, general-purpose programming language.

PyVISA	A Python package for support of the “Virtual Instrument Software Architecture” (VISA), in order to control measurement devices and test equipment for example via GPIB, RS232, Ethernet or USB interfaces [1].
Qt4	Qt4 is an open-source toolkit for creating graphical user interfaces and cross-platform applications.
ROI	Return on Investment
RS232	Recommended Standard 232. The standard for serial communication transmission of data. The current version of the standard is TIA-232-F, Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange.
SCPI	Standard Commands for Programmable Instruments is instrument command language for controlling instruments in a standard manner [2].
SMPS	Switched Mode Power Supply
SW	Software. In embedded systems software means any program executed on processor or microcontroller.
TestStand	National Instruments TestStand is a test management software for developing automated test systems.
UI	User Interface. Interface (screen, signal lamps, buttons, keyboard, etc.) provided for the user to interact with the system.
USB	Universal Serial Bus. Communication interface used in computers for interfacing embedded or external peripherals.
VISA	Virtual Instrument Software Architecture is an I/O API (Application Programming Interface) used for communicating with test and measurement instruments from a computer. The VISA standard [3] includes a set of specifications for communicating with resources of test and measurement equipment through their I/O interfaces. The IVI Foundation maintains the VISA standards.
WCET	Worst-Case Execution Time

1 Introduction

This study was made for a Company in Helsinki, Finland, which design and manufacture patient monitors, patient monitoring systems, monitoring parameter modules, and monitoring device accessories for hospitals. These products are used in hospitals for patient diagnosis and for monitoring patient condition during operations, transport and care through the whole hospital stay. Patient monitoring devices are used mainly in clinical environments like emergency care, operating room, intensive care, and recovery ward.

Parameters monitored from the patient are the basic vital signs and a set of more specialized parameters. Vital sign parameters are for example heart rate, blood pressure, breathing rate, and body temperature. Specialized parameters are for example inhaled and exhaled breathing gases concentration and flow, blood or tissue oxygen saturation, muscle relaxation level, brain activity and heart pumping flow rate and volume.

Patient monitors and systems development work is in principles like any electronics and embedded system development. Processors and microcontrollers are used to gather and analyze data and for controlling the system. Unique flavor to the development work comes from the medical device regulatory requirements and the responsibility to produce safe and effective devices.

In recent years, the Company has been investing lots of resources to develop state of the art, automated system testing methods and infrastructure. Automated system testing allows to test more and to test faster, and the existing resources can be used for challenging and exploratory testing. Currently, the automated testing system is up and running and development work to increase testing coverage for system software is progressing well. Now it has become time to think about how to include automation also for testing the system hardware interfaces. This study is part of the Company automated testing system development and a pre-study for adding features for hardware interface testing automation.

1.1 Objectives and Outcome

This study focused on the automated system testing of embedded systems and gathering information for answering the following questions:

- What kind of design defects are not discovered until at system level testing?
- Which hardware features or interfaces are important to test or monitor in automated testing of embedded system?

This study aim was to give recommendations on which hardware interfaces should be monitored, controlled or stressed and how it could be implemented in the automated system testing.

The results are used in the Company to plan future upgrades and improvements in the automated testing systems.

1.2 Document Outline

This document is divided into nine chapters.

The first and second chapters introduce the study subject, objectives and outcome and give an overview of the methods used in the study and the materials analyzed.

The third chapter gives an overview of the patient monitoring systems and discusses the specific characteristics of the system hardware and software development work.

The fourth chapter discusses automated system testing in general and explains the benefits of test automation. Also, two test automation frameworks already in use in the Company are introduced briefly and possibilities to integrate test and measurement equipment to them is discussed.

The fifth chapter introduces typical electronics design process and how the unit and block level testing can be made for hardware. This chapter gives a rationale why the system failures caused by defects, which should be typically found in the unit or block

level testing, may not require specific features in the system testing and automation of it.

The sixth chapter gives examples on two system-level defects which are difficult to find in the hardware unit and block level testing but could be revealed in automated system testing if suitable methods are available in the test system.

Chapter seven includes analysis of the data and information collected in the study.

The final chapters, chapter 8 and 9, has the recommendations for the automated system testing further development and conclusions.

2 Methods and Materials

The research methods in this study were a literature review and semi-structured qualitative interviews. Data from the literature and interviews were analyzed to find answers to the research questions and a basis for the recommendations.

2.1 Literature Review

Literature review had two goals. The first goal was to find examples of already established testing systems and methods for automated system testing of embedded system hardware. These examples were analyzed and evaluated, and the key information was taken as input to the recommendations produced in this study. The second goal was to find case examples of electronics and embedded system hardware design areas which can be challenging to test at the unit level and to how to stress and monitor them in the system level testing.

As a summary, the literature on automated testing of embedded systems was reviewed from the following perspectives:

- Why the system was built and what has been the benefits of it?
- What kind of tools has been used in test system development?
- What kind of failure modes are difficult to find without system level testing?
- Which hardware signals have been used in the case studies to identify the system is malfunctioning?
- How were the embedded system signals monitored?
- How has the data been gathered and stored?
- How has the system under test been communicating with the test system?

The literature search was limited to the following digital libraries and types of documents:

- Journal articles, research papers, and conference proceedings from:
  - IEEE Xplore Digital Library (https://ieeexplore.ieee.org)
  - ACM Digital Library (Association for Computing Machinery) (https://dl.acm.org/)

  - ScienceDirect database by Elsevier (https://www.sciencedirect.com/)
- Master's and bachelor's thesis from:
  - Open Repository Theseus – the theses and publications of the Finnish Universities of Applied Sciences (http://www.theseus.fi/)
  - Aalto University, Helsinki, Finland, library collections through the Finna.fi service (https://aalto.finna.fi/?lng=en-gb)
  - Tampere, Finland, University of Technology library collections through the Finna.fi service (https://tut.finna.fi/?lng=en-gb)

Due to a large number of publications available in the selected databases, the literature search was limited to a specific range of publication date and a set of keywords. The literature search was limited to publications within years 2008 – 2018. Publications were first searched with a full-text search using a specific set of keywords and logical operands available in the digital libraries search tools. The set of keywords and rules were selected based on an analysis of common terms in 6 papers, which were screened manually from preliminary searching results with specific keywords.

Depending on each of the digital library search tool features and logic, the search was executed either as 18 separate searches or processed as single search with logical operands. In any case, the 3+3+2 set of keywords resulted in 18 searches (3 x 3 x 2). Keywords and their combinations in the search are shown below.

```
("automated test" OR "automated testing" OR "automatic test")
     AND
("test system" OR "system testing" OR "system test")
     AND
(hardware OR "functional test")
```

From the search results, the document title and abstract were evaluated first. Based on the title and content of the abstract, a set of documents were selected for more detailed review. From the remaining documents, the full content of the document was reviewed.

A document was selected as an applicable reference to this thesis if it provided information on answering the questions set in this thesis and literature search objectives, and it presented a case study for a real-life embedded system test automation.

2.2 Interviews

Semi-structured qualitative interview of electronics and system design engineers of the Company was used to gather data on the following topics:

- What kind of hardware related defects has the expert faced?
- How was the defect discovered?
- Could the defect have been discovered in automated system testing?
- What kind of test equipment and signaling to test system would have been required to automate the test?
- Based on the expert's experience, what kind of measurement techniques should be considered to improve automated testing?

In the study five respondents were interviewed face-to-face for gathering the information, lessons learned and ideas on how to develop the automated system testing.

The interview respondents working careers were ranging from 14 to 44 years. On average the respondents had 27 years working experience from electronics design and manufacturing industry. Interviewed respondents had worked for example with medical devices, semiconductors industry, industrial electronics, and consumer devices.

The interviews used a framework and procedure shown in Appendix 1 – Interview guide. The interview guide was printed as a separate document and given to the respondents before the interview. The interview guide was also used as a framework in the interview to start the discussion and to lead the respondent to explain relevant cases and lessons learned from his/her career in the electronics industry.

2.3 Verification

Verification of the study output was arranged as peer review among the experts interviewed during the study and experts developing the automated testing system.

Ideally, the verification of the recommendations could have been made by prototyping some of the recommended methods with real hardware and measurement equipment. However, building prototypes were not in the study scope, as the operation of current automated system testing environment in production cannot be interrupted, and resources to build a new one was not available at the time of the study.

The peer review consisted of two parts. First, a preliminary summary report was written from the analyzed interview data and the literature review information. This report was sent to the experts for review. The second part of the review was a workshop with experts and other stakeholders. In the workshop, the preliminary summary report was presented and discussed. The workshops primary purpose was to gather feedback and get corroboration for presented ideas. A secondary goal of the workshop was to gather further data from the electronics, test automation and system design engineer's experiences on test automation of embedded systems. With the peer review feedback, proposals and new data from the workshop, the study report was finalized.

3 Patient Monitoring Systems Development and Test Automation

Patient monitors are computer-like medical devices, which integrate several processors, microcontrollers and various purpose-built analog and digital hardware blocks as a large embedded system. This kind of system consists of several communication interfaces, which are used to exchange information between the pieces of system software spread around the hardware modules.

The end user interacts with the system through a clinical application and can plug in or disconnect the peripherals and measurement modules. The clinical application shows the actual physiological parameter data and waveforms measured from the patient. Also, patient monitors are often connected to a hospital data network. Through the hospital network, the data, waveforms, and alarms are visible in the hospital ward control room. Network interface to control room can also be used for defining the patient monitor configuration and settings and for software updates.

Typical patient monitor with data acquisition devices and parameter measurement modules include approximately 5 – 20 processors or microcontrollers and several communications interfaces like USB, Ethernet, PCI and various proprietary and standard serial interfaces.

Figure 1 on the next page shows a typical patient monitoring system with peripherals, data acquisition devices, and measurement modules. In Figure 1 the actual patient monitor (1) runs an operating system with hardware abstraction layers and drivers and the clinical application. Patient monitor can have an external display (2) with a dedicated user interface (UI) functionalities. Data acquisition devices (3) (4) (5) are for connecting, collecting and multiplexing data from various measurement modules (6) (7) (8) for different physiological parameters. The data acquisition devices are typically from different generations of the system, but backward compatible with most recent clinical application. The same applies to actual measurement modules – the modules are provided for various measurement transducers, and sensors (9) developed in-house or by third parties over a long period. The connection to the hospital network (10) allows transferring data and alarms to control room or additional clinical applications for data analysis.

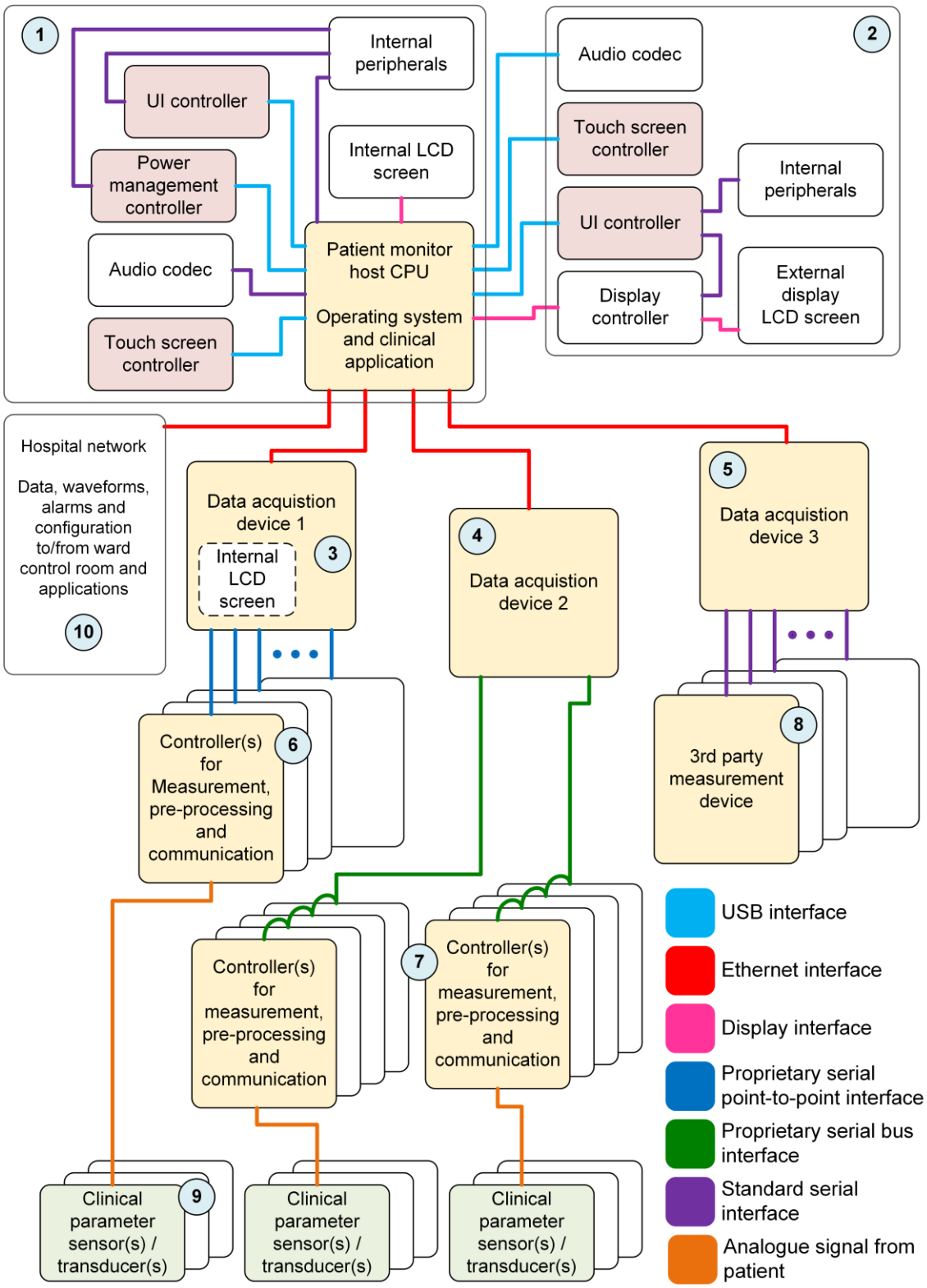


Figure 1. Example of a patient monitoring system and hardware interfaces in it

Medical device development is highly regulated. In Europe, the main requirement for patient monitors is medical device regulation (EU) 2017/745 [4]. The responsible organization developing medical devices must have a quality system to demonstrate the development work and operating procedures are controlled. The quality system is usually built in the companies to meet the requirements and guidelines of EN ISO 13485 [5]. Another requirement for the development procedures is the EN ISO 14971 [6], which defines how the risks of the medical device should be assessed. Requirements for safety, performance, and effectiveness of medical devices are defined in EN 60601-1 [7] and EN 62366-1 [8]. Also, there are several other standards which are setting requirements for specific products. As an example, the medical devices software development, in general, is controlled by the EN 62304 [9] and patient monitors have their own specific standard IEC 80601-2-49 [10], which define basic safety and essential performance requirements. Patient monitor product requirements and design specifications developed for a specific product model or software revision trace back to the regulatory requirements. Verification of the product functionalities against these product requirements and design specification is one of the most critical tasks in patient monitor development. Verification testing coverage and level of documentation must be comprehensive to be able to demonstrate all the regulatory requirements are met and the product can enter the markets.

Another unique characteristic of medical devices is their long service life and a large number of modern and legacy peripherals and accessories. The whole system may have been developed over a long period, and support is required for the legacy devices developed several years or even decades ago. Also, support and backward compatibility for legacy accessories and measurement modules is an integral part of the customer experience. Customer expectation and a regulatory requirement for medical device manufacturers are to gather information about problems and improvement opportunities and provide updates to the existing installed base of devices. For example in Europe, the medical device regulation (EU) 2017/745 [4] discuss in Article 10 Sections 9 and 10 [4, p. 24] the post-market surveillance, which in practice requires that device manufacturer shall provide updates and fixes. Also, the general safety and performance requirements in (EU) 2017/745 Annex I Chapter I Sections 17 and 18 [4, pp. 100, 101] sets guidelines for the medical device product development, verification, and validation. These requirements create a big challenge for system acceptance testing and verification. Any update or fix to the system may trigger re-verification and validation of the whole system.

To speed-up and to improve the quality of the verification and acceptance testing of new products and design changes, the Company has been investing a significant amount of resources in system test automation in recent years.

In medical devices and embedded systems, the software development methods have changed slower than in consumer device markets, but also with medical devices, the software is today developed using iterative and incremental development (IID) methods. With the IID methods, the software is developed in relatively short cycles and by using continuous integration or nightly builds and by utilizing automated testing. Software unit tests can reach a high level of automation and testing coverage and can be run for each software build. Also, the complete system level integration and acceptance testing can be automated by using software robots simulating real-life use scenarios and user interactions. The Company has been achieving well the above-listed targets with the current test automation system in place.

Medical devices electronics hardware (HW) development can also be made iteratively and incrementally, but the pace of these increments is much slower than in software development. Hardware development includes waiting times related to for example printed circuit board (PCB) design and manufacturing and the component assembly. Also, the level of test automation is usually lower in hardware development. Making fully automated testing systems, while the electronics are being developed, would cause the design iteration cycles to be even longer due to the similar waiting times related to the tester hardware ordering and building. Also, the cost of such automated testing would be very high. When the system electronics service life is long, and hardware iterations can differ from each other significantly, the return on investment on test automation is often not high enough to justify fully automated testing of hardware. Due to this, during patient monitoring systems development the hardware is often tested manually or with semi-automatic solutions on the block or interface level to make a compromise between testing coverage, development time and cost of test automation. Another reason for manual or semi-automatic testing is, that the mechanical form factor of the printed circuit board assemblies, connectors, and testing interfaces vary between different products, or may change between the product prototype iterations. Changes in the form factor make it more challenging to make a fully automatic testing system, which could be used through different device types and device generations.

System level testing stresses the system to the limits, and new problems and defects can be found both from software and hardware. As an example, the execution time and latencies from interrupt handling could cause failures in the system functionality. Another example is unexpected disconnection and reconnection of devices or sudden power loss. Recovery and restarting the system from different states may also cause unpredictable results.

Like software system testing, the hardware system level testing could be automated by using software robots simulating real-life use scenarios and user interactions and by recording information from the hardware at the same time. Also, for example, the peripheral devices connection and disconnection could be automated with devices controlled by the automated test system. However, it is not possible to include test equipment and controls for all hardware functionality automated testing. A balance between hardware test automation cost and benefits must be considered.

Currently, in the Company the system software and the automated system test software collect logs from the system under test, which can be used to verify the system functions correctly. These logs also help on debugging and finding the root cause for defects and crashes. However, as the level of test automation for hardware is low, there are very few data logged from the system about the hardware functionality. Also, the system does not stress the hardware or change the hardware configuration automatically.

The best return on investment could be found in the automated system testing by adding features for testing and monitoring hardware interfaces, which does not change between the hardware prototypes or product generations. This way the automated system testing infrastructure could be reused in future products and across the product portfolio.

4 Automated System Testing

This chapter discusses the benefits of system test automation, when to automate and what kind of tests should be automated. Also, two test automation frameworks are briefly introduced and adding hardware test equipment into them is discussed.

For an embedded system, manual testing is easy to start, and initial investment may be low. However, when the system size and number of features increase, and there is a need for repeated regression testing, manual testing may become a bottleneck for product development. At this point, the considerations on the benefits and return on investment (ROI) of test automation usually arise.

There are several benefits of automated testing. First, manual testing can be slow and prone to human errors [11, p. 463] [12, p. 302]. Automating the testing can make the testing repeatable and fast. Another benefit of the test automation over manual testing is that it enables types of testing which are not even possible to execute manually. This kind of tests are for example stress and load testing, timing tests and communication response time tests [12, p. 302]. Another benefit to consider is the re-use of the test cases. If the software is designed to be used in multiple platforms, test automation can reduce the time and cost of re-testing the software in new hardware platform [12, p. 302].

Most of the considerations for automated testing are related to the return on investment. The automated testing initial investment is usually higher and time to ramp it up is longer compared to manual testing. The test development in a test automation tool often looks like writing software and requires trained and skilled developers [11, p. 463] [12, p. 302]. Need for skilled developers increase the cost and makes it difficult to scale up quickly [12, p. 302]. Also, the maintenance of automated testing and procedures can be more complex, more expensive and time-consuming. That is because the system under test, software, test cases, and documents depend on each other and must be maintained for each change [11, p. 463] [12, p. 303].

The advantages and considerations as mentioned earlier apply very well for medical embedded systems as well. Medical embedded systems could be considered as a multiplatform system, even if the software is not typically released simultaneously to multiple platforms. Instead, new hardware platforms are introduced usually because some

component has become to end of its life and is no longer available. Typically, in hardware changes the system interfaces stay the same and new hardware and software are adapted to it and should be backward compatible. Interface backward compatibility means the already developed test cases addressing system internal or external interfaces could be easily re-used with refreshed hardware as well.

As a conclusion, compared to manual testing the automated testing initial investment and maintenance costs are higher and time to ramp it up is longer. However, embedded systems for medical applications tend to have a long service life, and customers expect updates to get more value for their investment. For a system which hardware does not change often, is modular and uses standardized interfaces, but which software evolves through-out the system lifetime, the return on investment of automating the system testing can be significant through faster regression and verification testing. Also, the overall testing coverage can be higher as some tests are possible only with automation and more tests can be run in the project timeline.

When a decision is made to automate system testing, it should be considered what kinds of tests are beneficial to automate. P. Tripathy and K. Naik [13] gives examples on types of tests cases which can benefit from test automation and have developed the following list:

> ***Less Volatile***: A test case is stable and is unlikely to change over time. The test case should have been executed manually before. It is expected that the test steps and the pass-fail criteria are not likely to change anymore. [13, p. 396]
>
> ***Repeatability***: Test cases that are going to be executed several times should be automated. However, one-time test cases should not be considered for automation. Poorly designed test cases which tend to be difficult to reuse are not economical for automation. [13, p. 396]
>
> ***High Risk***: High-risk test cases are those that are routinely rerun after every new software build. The objectives of these test cases are so important that one cannot afford to not re-execute them. In some case, the propensity of the test cases to break is very high. These test cases are likely to be fruitful in the long run and are the right candidates for automation. [13, p. 396]
>
> ***Easy to Automate***: Test cases that are easy to automate using automation tools should be automated. Some features of the system are easier to test than other features, based on the characteristics of a particular tool. Custom objects with graphics and sound features are likely to be more expensive to automate. [13, p. 396]

> ***Manually Difficult***: Test cases that are very hard to execute manually should be automated. Manual test executions are a big problem, for example, causing eye strain from having to look at too many screens for too long in a GUI test. It is strenuous to look at transient results in real-time applications. These nasty, unpleasant test cases are good candidates for automation. [13, p. 396]

> ***Boring and Time Consuming***: Test cases that are repetitive in nature and need to be executed for longer periods of time should be automated. The tester's time should be utilized in the development of more creative and effective test cases. [13, p. 396]

As a summary, automating everything on the system testing may not be feasible. However, when acknowledging that medical embedded systems re-use of interfaces is common and backward compatibility to legacy peripherals is often a must, investing in test automation of system software and including the modular hardware to the test cases can be beneficial.

4.1 Test Automation Frameworks

Test automation framework means a test automation infrastructure consisting of test tools, equipment, test scripts, procedures, and people needed to make test automation efficient and effective [13, p. 400]. Test automation infrastructure should ensure:

- Different test tools and equipment are coordinated to work together.
- The library of the existing test case scripts can be reused for different test projects, thus minimizing the duplication of development effort.
- Nobody creates test scripts in their own ways.
- Consistency is maintained across test scripts.
- The test suite automation process is coordinated such that it is available just in time for regression testing.
- People understand their responsibilities in automated testing. [13, p. 400]

Test automation framework shows its efficiency, especially in regression testing. Regression test is typically a collection of test procedures which are run on software test release. Regression test procedure tests for any unintended changes to the software's behavior that are unrelated to the specific changes made between test release versions [12, p. 300]. Regression test sets can also be run to verify a specific change in system hardware. Being able to re-run a test set for a specific interface or whole system fast will accelerate product development.

4.1.1 Robot Framework

One test automation framework in use in the Company is the Robot framework. Robot Framework is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD) [14]. Robot Framework is an open source project and inspired by the master's thesis [15] project of Pekka Klärck (Laukkanen) [16, p. 26].

Robot Framework uses a keyword-driven testing approach. Its test libraries can be implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases [14].

Robot Framework test cases are human-readable documentation stating the intent of the test. The test case is made up of keywords forming the syntax of the test case file. The keywords can be provided as standard by the Robot Framework, or through an imported or self-developed library [17, p. 116].

4.1.2 National Instruments TestStand

Another test automation framework the Company uses is commercial software called TestStand by National Instruments. TestStand is test management software that includes a ready-to-run test sequence engine that supports multiple test code languages, results reporting, and parallel multithreaded testing. [18, p. 85].

TestStand is well suited for hardware testing, as it can call and run executable tests written in National Instruments LabView programming language. LabView is a common tool used in HW testing, as there are compatible instrument drivers available for a majority of the commercial test- and measurement equipment like oscilloscopes, power supplies, spectrum- and network analyzers and HW test automation switches and multiplexers. National Instruments is one of the market leaders in test automation software [19]. Examples of available drivers can be found in the National Instruments service called Instrument Driver Net (IDNet) [20].

4.1.3 Integrating Test and Measurement Instruments to Existing System

The Company uses already the two frameworks as mentioned earlier. The Robot Framework is used mainly for system software test automation. National Instruments TestStand is used almost exclusively for the final product and HW component functional testing in production. Some hardware unit testing is made with National Instruments LabView stand-alone programs.

Adding test and measurement equipment for embedded electronics system testing would require integration of a set of measurement and control equipment through their control interfaces to a computer handling the test automation. The integration could be made by selecting test and measurement instruments which vendor provides VISA (Virtual Instrument Software Architecture) compliant application programming interface (API) to the device. This interface is supported directly by the National Instruments LabView, which is typically used to write tests for TestStand. Measurement equipment interfacing, instrument drivers and for example hardware abstraction layers are explained in detail in the National Instruments application note [18, p. 97]. Also, the API's could be used directly through Python programming language with the open source package called PyVISA. PyVISA would enable controlling the measurement equipment from software written with Python [21, p. 5], which is used in the Robot framework. Example of using open source software and PyVISA is for example in [22] and [23]. Older instruments using SCPI commands for example through IEEE-488, RS232 or USB could be used as well. SCPI commands are ASCII strings and can be integrated into the automated test system as long as there is a suitable hardware interface in the controlling computer.

The Robot framework and National Instruments TestStand could be made to communicate with each other. Communication between the two tools could be a solution for a case where a suitable measurement instrument or test device driver is available only to National Instruments LabView.

As a conclusion, the tools in use in the Company already enable integrating the test and measurement equipment to the automated test system. Expanding the system test automation to include hardware controls, measurements, and monitoring requires hardware engineers to specify the tests for test and measurement equipment which can be integrated into the test automation tools in use.

5 Electronics Design Process

This chapter gives a briefing to the electronics design process in embedded system development. The design and testing activities during the electronics hardware design are explained with a practical example. Also, the purpose of this chapter is to give an overview of what kind of design errors or defects could be revealed and fixed separately from a fully integrated system.

The design process typically includes checkpoints or milestones which define a standardized or recommended set of tests to verify the design functionality and maturity. The different methods of testing used for a block or feature verification are discussed. In addition, electronics component reliability and failure mechanisms are introduced briefly.

Appendix 2 gives an example of an embedded system product development process. This example is based on the authors 20 years of experience about best practices from electronics and product development projects. In this example, the actual development work happens in the product design phase. The product design phase is built around prototypes. Each prototype allows new features or functionalities to be added to the design and refinement of the unit and system requirements. One important role of the prototypes is to allow testing and verification of the functionalities apart from the whole system. Prototypes in hardware development can be compared to software releases in the iterative and incremental design process. Each iteration increases the maturity of the design and allows the verification and validation of the individual parts. At the same time, product development advances towards the full system integration and product release. Release of the product may happen when each interface, feature, and functionality has passed testing against the requirements and system is functional.

As the embedded system includes processors or microcontrollers, the software is a vital part of the system. Hardware development process and prototype building is often adapted to available tools and applied software development methods. Mikko Kerttula gives examples on these relationships and tools on the VTT PUBLICATIONS 615 [24, pp. 40-71]. Ideas about the hardware and software co-design are also presented in several papers. As an example, Timo Punkka makes an important note about the possibilities to test parts of the hardware units without the full system. Semi-automated testing can be used to test the hardware-software interfaces with simplified software

drivers for hardware [25]. Lima et al. make a similar recommendation – developing the embedded software and the hardware together and synchronized with each other enable the hardware unit testing [26] [27].

With the techniques and examples as mentioned above, the hardware development cycle including a prototype build could give relatively good testing coverage for hardware blocks, interfaces, and functionalities. Figure 2 (p. 20) shows an example of the design activities from electronics hardware development point of view for one prototype iteration. Figure 2 identifies four areas where the hardware or system verification and testing could be made.

Part of the design verification could be made through schematics design reviews or by simulation (Figure 2 (1)). Simulation of functional blocks is usually very straight forward. For example, simulation of DC-operating points, circuit transient or frequency responses can be made in modern electronic design automation tools. Design review could include for example studying the reliability and de-rating of the selected components in the expected operating environment and conditions.

When the schematics are considered mature enough, printed circuit board design can be started. Like schematics, PCB review and simulations (Figure 2 (2)) can be used as part of the design verification. From the printed circuit board design typically at least the signal integrity is verified by simulation, and most electronics design automation tools have functionalities for that. Additional simulations could be thermal and mechanical simulations to see if the components on the printed circuit board can run cool enough and they stay on the board for example when the product is dropped.

When the prototypes are available, block, feature and interface testing (Figure 2 (3)) is usually the next step. At this phase actual measurements can take place and finding circuit operating points, exploring design margins and confirming the simulation results is possible. When the prototypes are mature enough for system integration, and there is system software available, system testing (Figure 2 (4)) could start. At system testing, any testing applicable to the whole system can take place, and system level interfaces functionality is tested. Also, from the electronics design point of view finding design margins against environmental and regulatory requirements is usually one of the vital areas of testing.

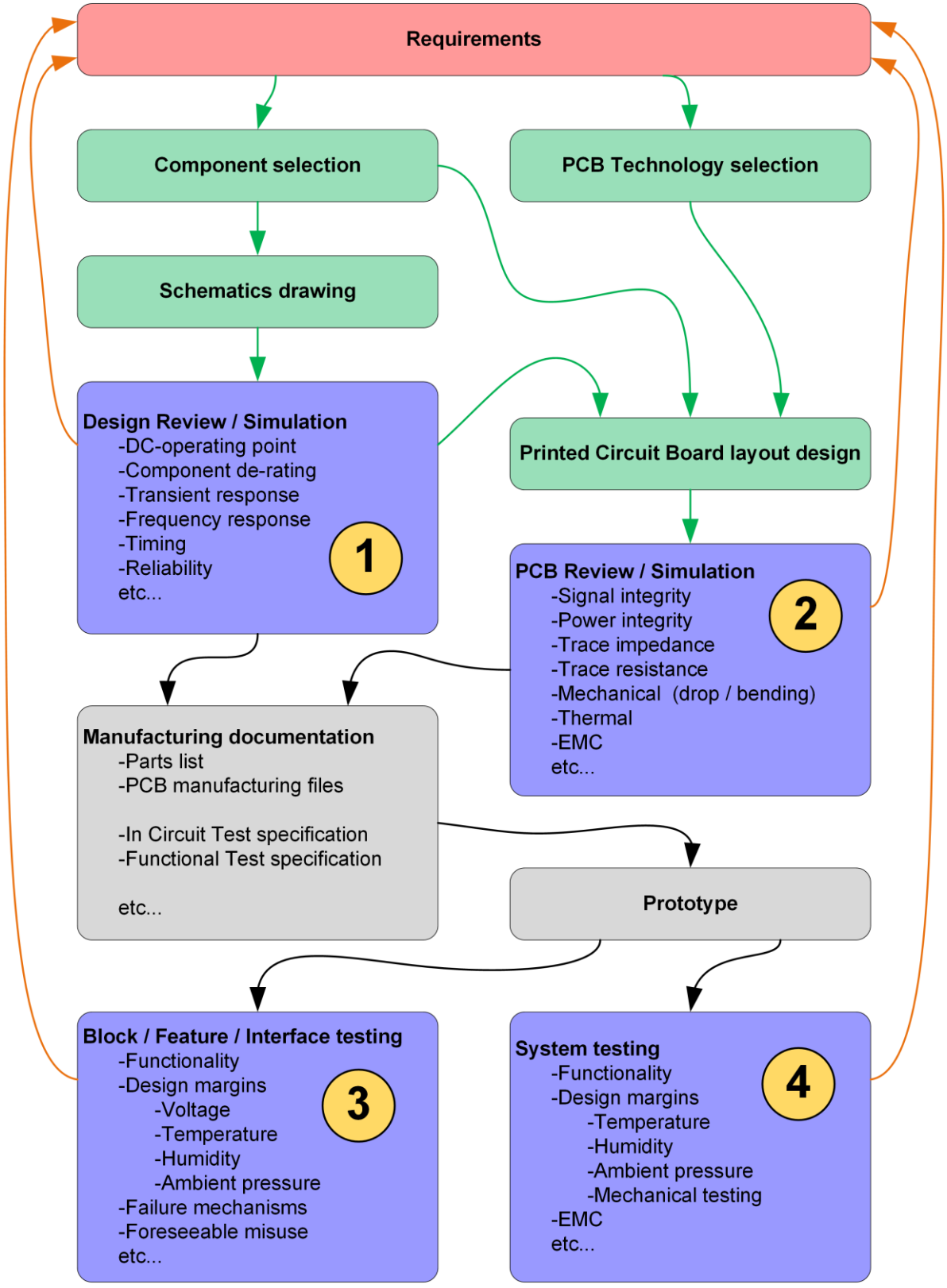


Figure 2. Example of activities during hardware prototype iteration

To further explain the idea of hardware unit testing, a practical example can be used. Figure 3 shows an imaginary switched mode power supply (SMPS) and the control interfaces to a microcontroller (MCU). This hardware unit can be used to explain the reviews, simulations, and testing shown in Figure 2. In this example, the SMPS could be a part of an embedded system, which has a system level requirement to synchronize the SMPS to dedicated clock source and possibility to fine-tune and monitor the output voltage level in different system use scenarios. The SMPS would be started automatically when input power supply V_{IN} is available. At the start-up, the SMPS is synchronized to its internal oscillator, and the output voltage V_{OUT} is set with external resistors. The embedded software can adjust the SMPS operating frequency, and output voltage with two timer outputs PWM_{OUT1} and PWM_{OUT2}. The output voltage can be monitored through the microcontrollers integrated analog to digital converter input ADC_{IN}.

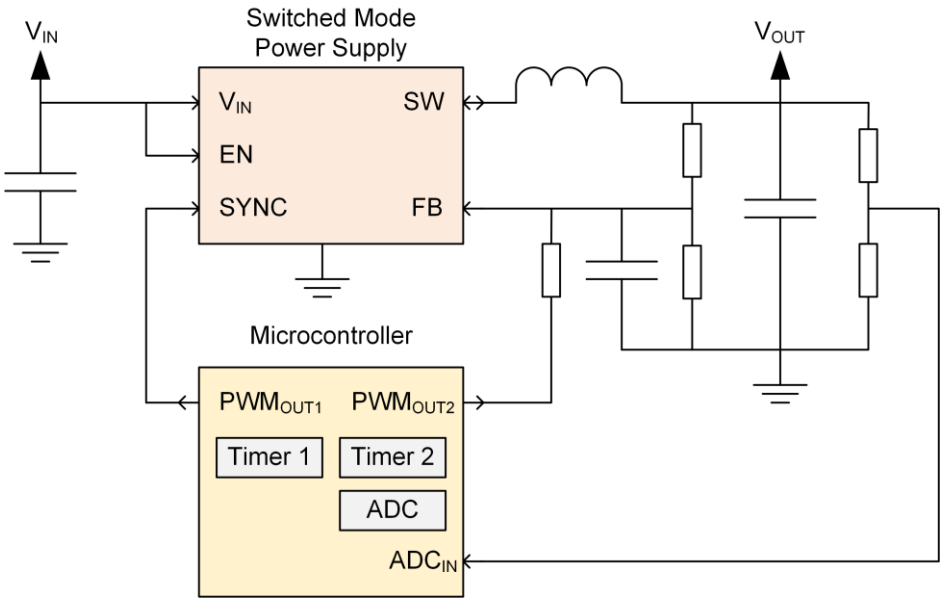


Figure 3. System controlled and monitored switched mode power supply

The HW team would start the design by selecting suitable components and drawing schematics for the HW unit and the interfaces to the system. Based on the selected components and system interfaces the requirements for the hardware and software can be defined. Software requirements would be the microcontroller timer output signals PWM_{OUT1} and PWM_{OUT2} frequency and pulse ratio range and the ADC_{IN} input range and scaling. With these requirements, the software team could start developing the necessary drivers and control logic.

The first prototype of this HW unit could be an evaluation board of the SMPS from the integrated circuit (IC) manufacturer. IC manufacturers usually provide simulation models and comprehensive reference design data and other technical data. These first steps correspond to steps (1) and (2) shown in Figure 2.

The evaluation kit can be modified to include the selected external components. With the modified evaluation kit, the HW development team can check the power supply parameters like efficiency, line and load regulation, load transient response, electromagnetic noise, short circuit protection and operation over ambient temperature. Also, the control signals for SMPS frequency synchronization and output voltage adjustment can be trialed out by using signal generators. From all the input gathered the requirements for HW and SW could be refined. These steps correspond to step (3) shown in Figure 2.

The next prototype could be the first release of the system board, which integrates the SMPS and microcontroller on the same printed circuit board. For this electronics assembly, the same tests which were made with the SMPS evaluation kit can be repeated. Also, all the testing and verification steps 1-4 shown in Figure 2 can be included for this prototype. Furthermore, this prototype can be used to test the software controls. The software, in this case, can be just the HW drivers which parameters can be adjusted from some control interface. Full system level software is not needed. If new prototypes are produced, the testing can be limited to changed parts of the design. Again, after these steps, the unit and system requirements can be refined, and the new prototype can be produced if necessary.

With the prototyping and testing of the HW and SW units the testing coverage of this example SMPS can be very high. The HW and SW units can be integrated later to the whole system and verified at the system level. System level testing can concentrate only on the system integration related topics, as the unit testing has covered already both the HW and SW basic functionalities.

During the HW unit development and testing also the single component and complete electronics assembly reliability can be evaluated. Component and assembly failure rates can be estimated for example with methods described in MIL-HDBK-217 and Telcordia SR-322. MIL-HDBK-217 is a military handbook about reliability prediction of electronic equipment and published by the United States of America Department of

Defense. The latest revision of MIL-HDBK-217 is F published in 1991, and there have been minor additions in 1992 and 1995. This document gives a part failure rate model and reference constants for various component types and the stress they are under in the design. Telecordia SR-332 describes similar component and assembly reliability prediction method as the MIL-HDBK-217, but with models and constants adjusted to fit better to commercial telecommunications products [28, p. 11]. The latest version of the SR-332 is the Telcordia SR-332 Issue 3 from 2011. These traditional methods give a single value called Mean Time Between Failures (MTBF) for repairable systems and Mean Time To Failure (MTTF) for nonrepairable parts. The MTBF or MTTF value should be used with caution, as they do not model for example realistic operating stresses and impact of manufacturing variation on reliability [28, p. 15]. Another thing to understand from MTBF is, that the failure times used to calculate the MTBF are assumed to be exponentially distributed and that the components of a system have inherently constant failure rates [29]. This may not be true with all components, and more detailed physics of component failure should be understood.

In embedded system development, a distinction should be made between failure due to the physical phenomenon and electronics functional failure. The statistical reliability estimates concentrate on physical failure of the device. However, instead of trying to estimate the product or system physical failure rates through statistics, a method called Highly Accelerated Life Testing (HALT) could be used instead. The HALT method can be used to find weak points causing physical failures, but it can also reveal underlying functional failures if the system functionality is monitored during the test.

HALT method goal is to stress the product beyond expected operating conditions and specifications until it fails. The stress may be for example rapid temperature changes, random broadband vibration, operating voltage, and clock frequency variation or a combination of them. Based on the failed unit analysis the product design margins can be seen and to discover the products operational limits [29]. Gray and Paschkewitz give good examples of failures which may be detected with the HALT method. As an example, cracked Ball Grid Array (BGA) package soldering may have a 100% fracture across the ball, but the surfaces without stress make contact and product may operate normally. However, under thermomechanical stress or vibration the conduction path can open, which results in a detectable failure [29, p. 32].

The HALT method can be useful for detecting electronics functional failures as well. When a printed circuit board assembly is manufactured, there is variability on the assembled components and assembly method itself. The variations can lead to timing and signal integrity failures [29]. The timing and signal integrity failures are seen typically as random software crashes or corrupted data. Writing software for a set of stress tests for digital data interfaces and running them on multiple samples over wide environmental conditions can be used as a unit test to reveal underlying timing and signal integrity problems. Including HALT testing to the block, feature, interface and system testing activities shown in Figure 2 can reveal underlying design defects before the actual full system integration and testing starts.

As a conclusion, the electronics product development process can include many useful practices to find design defects, and failure root causes already at the unit or functional block level. General good practice in electronics design is to test each functional block thoroughly and beyond the whole system environmental operating conditions. Confidence in the unit makes it easier to perform the system integration and reduces need for concentrating on the individual functional blocks in the automated system testing.

6 Challenging Items with Embedded System Functional Testing

The previous chapter, 5 – Electronics Design Process, explained how the electronics circuit units or functional blocks could be tested before the actual system integration takes place. Even if relatively high testing coverage and confidence in the different units can be achieved before system integration, there will be some testing which typically requires the fully functional system hardware and software.

This chapter gives two examples on embedded system design which can be challenging to test early in the product development phase, and before the whole system testing is possible. These two items discussed are the system level power integrity and interrupt and code execution timing variability. Purpose of this chapter is to introduce the topics and explain why the system level testing should have mechanisms to test and monitor the functionality of the hardware.

6.1 System Level Power Integrity

Power integrity means the power supply network ability to guarantee the quality of the supply voltage. Supply voltage quality specification typically includes parameters like DC voltage level, the accuracy of the DC level and allowed maximum and minimum voltage drop and overshoot. These parameters are given in specific steady state and transient load current situations.

Each functional block or feature in electronics can be verified quite well separately. Voltage regulators can be tested independently from the system for line and load regulation and line and load transients. This type of stand-alone verification can be made against the given supply voltage quality specifications of every component in the system.

In addition to testing of prototype circuits, the power distribution network (PDN) can be modeled, and voltage drops in the power supplies can be estimated from the PDN impedance and expected load transients. Also, a target or design requirement for the PDM impedance can be set when the minimum and maximum voltage the load tolerates are known. Target impedance is a concept widely used in embedded systems and microprocessor power integrity design to characterize the power distribution network

[30, p. 8] [31] [32] [33]. Target impedance can be a fixed value, or it can be a function of frequency [31] [33].

Figure 4 shows an example of the level of details when modeling the power distribution network from the voltage regulator to the silicon die, which is generating the changes in the load current. System simulation model could include a voltage regulator model, printed circuit board and de-coupling capacitors and the actual load, which in most of the cases is an integrated circuit like a microprocessor. Figure 4 shows these three parts.

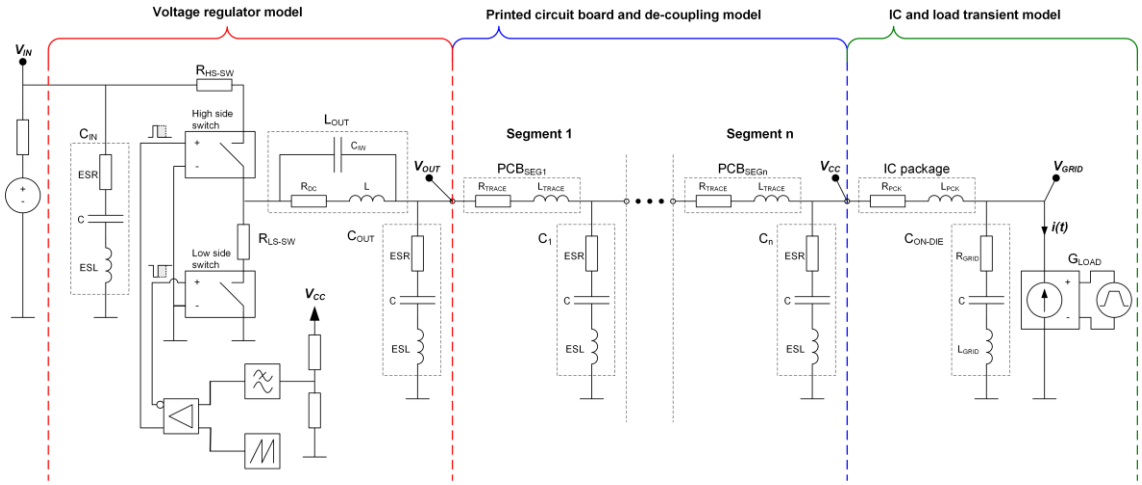


Figure 4. Example of a power distribution network (PDN) modeling details for power integrity simulations.

The voltage regulator modeling is quite straight forward in case a regulator IC vendor provides an accurate functional model. Also, the model can be based on load regulation and transient measurement results. Model of the voltage regulator should consider the resistance of the components and provide realistic line regulation with the varying load current. Also, the voltage regulator model should include realistic behavior for the load transient response. Load transient response means how well the voltage regulator can keep the desired output voltage constant when the load current changes. Voltage regulator model also includes the output bulk capacitor, which should be modeled with its parasitic elements like effective series resistance and inductance.

The printed circuit board and de-coupling model should include all the elements which effect on the impedance of the path from the voltage regulator to the load. The printed circuit board model is often extracted from a real board design with a 3D electromagnetic modeling tool, which output can be for example a multiple port scattering parameter (S-parameter) model. The printed circuit board model is combined with the de-coupling capacitor models and the impedance of the printed circuit board, and de-coupling capacitors between the voltage regulator model and the load can be calculated.

The last part of the model is the load. In case of an integrated circuit, the model should include the package model, silicon level decoupling capacitance and the load current. Unfortunately, these are usually not available and has to be estimated for example from the processor or functional block operating clock frequency and maximum power consumption values from the datasheet.

When the whole system level simulation model is ready, the power distribution network impedance and voltage at the load can be simulated. Figure 5 (p. 28) shows an example of the power distribution network impedance. The parts of the model shown in Figure 4 impacts the PDN impedance on different frequencies. The voltage regulator has limited load transient response, it is far away from the load, and it has large output capacitors. These factors make the voltage regulator effect on the power distribution network impedance dominant on the lower frequencies. Printed circuit board design and de-coupling capacitors effect on the power distribution network mainly on the middle frequencies. Printed circuit board layout and de-coupling capacitor placement can be optimized through the power distribution network impedance and to provide as low and flat impedance as possible over a wide frequency range. The integrated circuit package and silicon die level de-coupling dominates on the high frequencies.

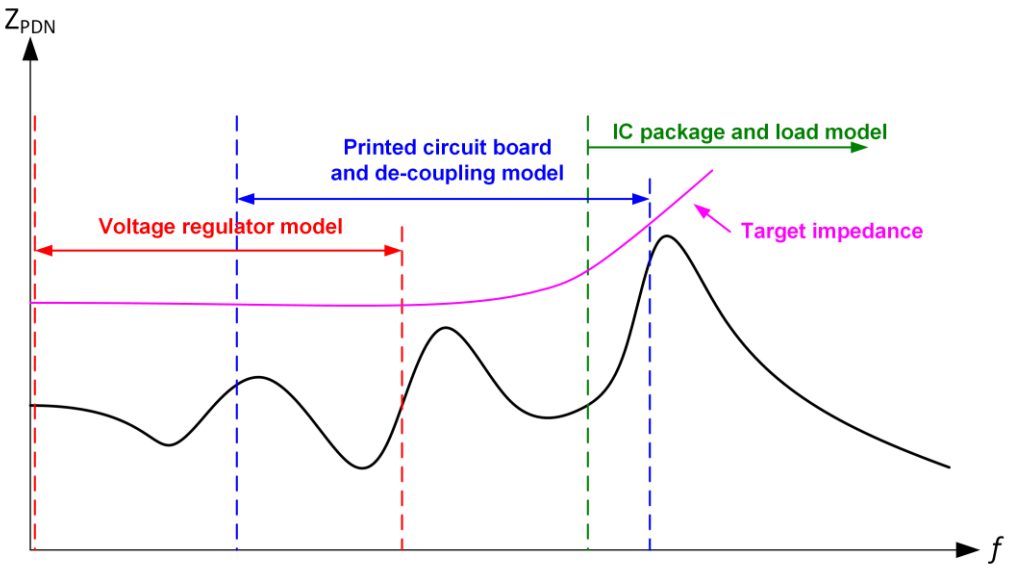


Figure 5. Example of power integrity simulation results for a power distribution network (PDN) impedance and voltage drop due to the load current.

Unfortunately, the actual load current transients are in practice impossible to predict accurately. Load current drawn from the power distribution network is typically a filtered version of the instantaneous load representation, and the current is averaged over time due to load changes throughout the workload activity and the operation of the units in the processor [30, p. 148]. The load current and transients a processor generates can be very complex and is highly dependent on the activity of the processor and system operation [30, p. 150]. The same applies to embedded systems as well - the load current may vary a lot and depends on the software being executed. In addition to single processor loads, in an embedded system, there typically are multiple other loads connected into the same power distribution network. Also, in this case, it is usually impossible to model accurately how all the loads behave together. Without accurate load transient models, even the most accurate modeling of the PDN and simulations gives estimates what the voltage drops may be in a real system. In addition, there is variability between integrated circuit samples and components of the power distribution network. Due to these facts, the embedded system power integrity problems could be revealed in a very late phase of the product development and in the worst case the end user discovers them.

Typical defect seen at system level due to power integrity problem is a sudden crash or halt of the system. If the processor or memory supply voltage drops too low because of a load transient, the data in the processor registers or memory can be corrupted or for example, some internal signal state is interpreted wrong. Data corruption or misinterpreted signal state may happen randomly and finding the root cause can be very challenging.

System testing should have mechanisms to address the unknown factors of the system power integrity. Ideally, the testing should stress the system and generate the worst case and varying load conditions. These conditions could be created in automated system testing, and the stress tests could be repeated to multiple hardware samples to find the variability between them. In addition, in regression testing, the stress test cases should be run by default to see the software changes have not created new defects related to system power integrity.

6.2 Interrupt and Code Execution Timing Variability

Another example of an item which is often impossible to test as a unit test is the system event timings. Individual interfaces usually work fine as they have built-in hardware signals or software protocols to synchronize the data flow. These can be tested separately and verified early in product development. However, in an embedded system, there can be several microcontrollers and processors which all can have multiple hardware peripherals. At the system level, all these must work together, and events triggered by each other should be handled keeping in mind the timing and synchronization. This chapter explains through a practical example the impact of interrupt and code execution timing variability on system functionality.

One basic functionality of the microprocessor or controller is to stop currently running code execution and perform a predefined task in case an interrupt signal is received. An interrupt could come from an external device, microcontroller or processor internal peripheral or another software process. When an interrupt is received, it is served by executing a predefined software routine related to that specific interrupt.

Embedded system software design usually assumes the interrupt is served within a specific time window after the interrupt is received. However, the time window can be defined only for a case when the processor is running on low load and is not already executing an interrupt service routine. If multiple interrupts are received at the same time or one interrupt is being served, and new interrupts are received simultaneously, there can be variability in the latency and order the new interrupts are being served. Multiple interrupts waiting to be served can also be prioritized, which again causes additional variability on the lower priority interrupts. Discussion and examples of the magnitude of the interrupt serving timing variability in different system configurations have been studied for example in [34] [35].

Depending on the software and system implementation, the interrupt serving time variability and latency may cause unexpected behavior in the system. Naturally, the system should be built in a way that it tolerates the variability in timing.

An example of a sequence of events which would require strict timing is an analog to digital (AD) conversions and control actions made based on the conversion result. Such a structure could be for example an ultrasonic flow meter and a proportional valve

controlling gas flow in an embedded system illustrated in Figure 6. In this example, analog to digital conversion request could be triggered by a timer inside a processor, which appears as an interrupt. The processor would have to initiate the AD-conversion, read the conversion result when AD-converter is ready, calculate the value for desired control action and send the new control values through a communications interface to DA-converter. In this sequence of events, there can be several interrupts to the processor. First, there is the request to perform AD-conversion, then an interrupt from AD-converter signaling the conversion is ready, then several interrupts from serial port peripheral while transferring the data from AD-converter to the processor and finally interrupts from another communication interface while communicating the new control parameters out. If the requirement for control loop accuracy and speed are not strict, the latency and variability in serving the interrupts may not be a problem. However, often this kind of control loops must be isolated to separate microcontroller or programmable logic to guarantee low enough latency and predictable variability in the control timing. Also, additional co-processor could be added in the system to handle servicing of hardware interrupts [36]. If the processor running the example control loop also performs other tasks, it would be important to run system-level acceptance testing and regression testing after any change which might affect the timing and latencies of serving interrupts. The tests should cover typical system use cases and extreme stress cases to verify the parts of the system requiring strict timings behaves as expected.

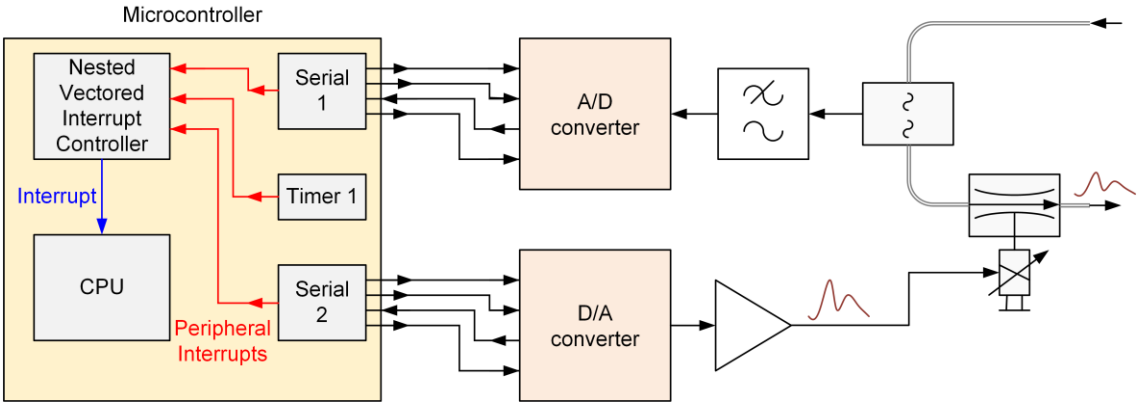


Figure 6. Example of gas flow measurement and control with microcontroller and interrupts to CPU.

Like interrupt servicing latency and timing variability, the processor architecture, where the code is being run, can also affect the system behavior. If the system software is running multiple threads, the processor architecture features like number of processor

cores, type of memory (static/dynamic random access memory, instruction/data cache and scratch pad or tightly coupled memory), pipelining, out-of-order execution, branch prediction, and speculative execution may cause variability in the code execution timing and latency [37, pp. 37, 70, 105]. Especially if the system requires a hard real-time operation, the details mentioned above must be considered. It may also be a system or even regulatory requirement [37, pp. 6, 7] to simulate, calculate or measure parameter called Worst Case Execution Time (WCET), for tasks and events requiring hard real-time execution. The embedded system behavior should be predictable and consistently execute the software with known latency and timing variability.

As a conclusion, like with the system level power integrity, also the interrupt servicing and code execution timing variability are in many cases unknown factors in the system. Because of this, system testing should have mechanisms to stress the system and generate the worst case and varying load conditions. These conditions are easiest to be created in automated system testing. Regression testing should include the stress test cases, and they should be run by default to see the software changes have not created new defects related to system timing and synchronization of events.

7 Data Analysis

7.1 Literature Review on Automated Testing Systems.

A literature review was conducted using the method described in chapter 2.1. As an example of the number of papers screened from one source, Figure 7 shows the results found from Elsevier ScienceDirect after each literature search process step. Searching with only one keyword or term, “automated test”, and limiting the results to publication years from 2008 to 2018 produced 1743 search results. When applying the whole set of keywords and rules defined in 2.1, the number of results was reduced to 207. Out of these 207 papers, the number was reduced to 9 when the title and abstract were reviewed. Finally, after reviewing the content of the remaining nine papers, two papers were considered useful and applicable to this thesis. The same process was repeated for the other sources specified in 2.1.

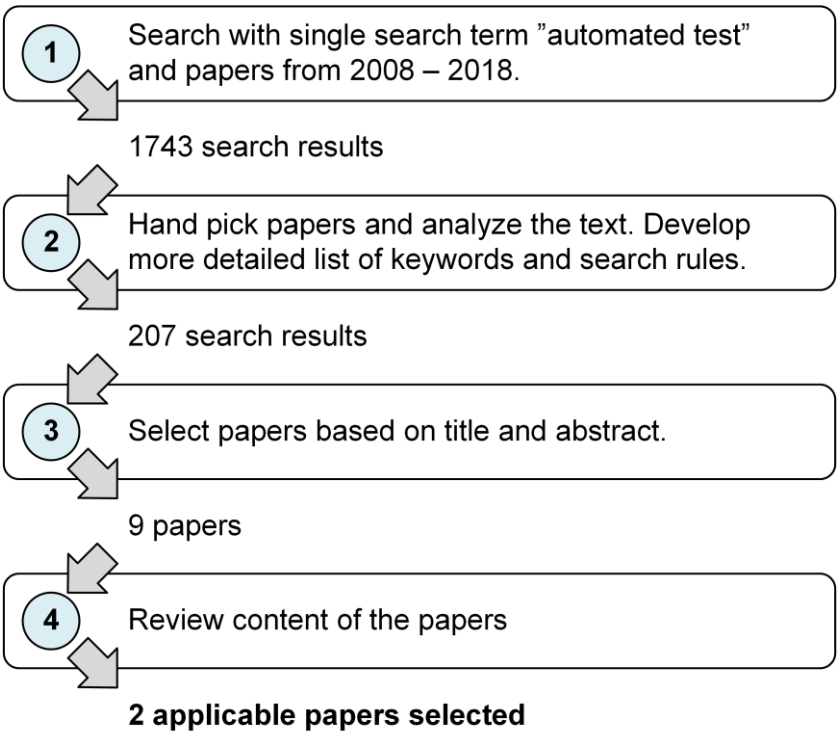


Figure 7. Number of papers selected from Elsevier ScienceDirect after search and review process

Similar numbers for IEEE Xplore digital library search were 889 papers with single search term “automated test”, 281 papers with a full set of keywords and logical operands, 37 picked based on title and abstract and finally four papers were selected for this literature review. From ACM Digital Library one paper was selected. From the collections of Theseus, Aalto University and Tampere University of Technology there were several thesis project reports on software test automation or functional testing of hardware. However, no paper was found applicable for this literature review as they did not describe a full embedded system test automation solution.

It was challenging to find papers which meet the set criteria for the literature to be reviewed. There is an overwhelming number of papers presenting methods for simulating different types of systems, and about system testing by using simulation models to replace parts of the system under study. Also, there is a large number of papers presenting methods for simulating a single functional unit or combined system software and hardware-in-the-loop simulations for verification. These papers were excluded from the review as they did not present case studies of whole system testing automation or did not disclose information about the equipment or methods used for hardware monitoring. Instead, they can be considered as examples of unit testing cases.

There were no papers found discussing specifically the system testing challenges for finding hardware failures underlying or left undetected from unit or block level testing. However, there were some excellent papers found describing whole embedded system test automation infrastructure and how it had been used for production, system integration, and acceptance testing. A short list of the papers reviewed is shown in Table 1 (p. 35). Chapters 7.1.1 – 7.1.4 have a summary of the papers listed in Table 1 and discussion about the methods used in the presented systems and how they could be utilized in the Company, for developing the in-house automated testing system further.

Table 1. Papers selected for review from the literature search.

Document reference number	Document title	Main topic of the case presented
[23]	Server Power Supply Test Automation Approach	Automated test system with open source tools (Python, PyVISA) for power supply characterization and production testing.
[38]	An automated environment for hardware testing using PXI instrumentation and LabVIEW software	Automated test system for embedded system functional testing in production. The system built around the LabView application and MySQL database.
[39]	A system for automated testing in the development of measuring devices for industrial process instrumentation	Complete automated test system with test result analysis and reporting tools written in C++ and with Qt4 for industrial control devices.
[40]	Simultaneous functionality verification system of multiple Set-Top Boxes	A testing system where the output from multiple systems under test are compared real time to the reference system.
[41]	Implementation of System Testing Automatization on Computer Aided Systems for Hardware and Software	Automated system, regression, and production testing system for marine echo sounder implemented with purpose-built software and hardware emulators.
[42]	Automatic test equipment for avionics Electro-Mechanical Actuators (EMAs)	Test system with real-time requirements for avionics actuators for gathering real-time data from the system under test and evaluating its response to the test system generated stimuli.
[43]	Development on electrical system performance test stand for combine harvester	Production testing system capable of characterizing the quality of the electrical installations of a combine harvester.

7.1.1 Use of Open Source, Commercial and Self-written Control Software

The following three systems presented in [23], [38] and [39] were picked as examples of how the automated test system control software could be implemented. The three example solutions are using either an open source software as much as possible and modifying it for own purposes, using fully commercial software and tools, or to write the drivers and test sequencing and reporting routines in-house. All three options have their benefits, and the authors of the selected papers have explained why the solution was selected.

In the paper [23] by Apostolaki-Iosifidou, Ramachandran, and Dong an automated testing system for characterizing a server power supply is presented. The system built in this case study is straightforward. Commercial off the shelf programmable AC/DC power supply, programmable load, and an oscilloscope has been connected to a computer through USB interfaces. The computer is running a script for measurement equipment control, data acquisition, data analysis, and reporting. Even if in this case study no real embedded system is being tested, the paper is interesting to this thesis study as it uses open source tools and particularly Python. Paper gives an example of how Python scripts and PyVISA instrument libraries have been used with the test and measurement equipment. PyVISA libraries enable communication with the test and measurement equipment from the Python script with SCPI commands (Standard Commands for Programmable Instruments). Testing equipment used in the case example is programmable power supply, load, and oscilloscope which all are connected to the controlling computer with USB interface. Measurement results from the system are voltage and current waveforms which can be presented in a time domain or as a function of load. Results have been stored in the system as text files. Authors motivation for building the presented system seems to be to show proof for the concept of using open source tools for system test automation. The method presented in this paper could be extended to cover larger embedded systems and communication interfaces of it, yet still at the same time using test and measurement equipment for the system under test hardware monitoring. The method is directly applicable to the Python-based Robot Framework in use in the Company.

The paper [38] by Ćatić, Lukić, et al. shows a production testing system for a wireless audio recording device. The system under test includes one microcontroller which handles audio recording and playback. In this example case, test and measurement equipment and software development tools are almost exclusively from National Instruments. Test system hardware presented in the paper has been built around PXIe backplane and chassis. PXIe or PXI Express is an adaptation of PCI Express bus to the PXI form factor. PCI or Peripheral Component Interconnect bus is widely used in personal computers, servers, and embedded systems for add-on cards and system components. PXI stands for PCI eXtensions for Instrumentation (PXI) and is a common form factor and backplane bus for modular test and measurement equipment. The test system application software has been developed with National Instruments LabView, which offers hardware abstraction layer drivers for the test and measurement equipment. The test system application in the example case stores data to the MySQL data-

base, which enables generating test reports with different templates. The authors compare the system to manual testing and state that "*proposed solution is up to 20 times faster, significantly cheaper and much more reliable than manual testing*". This example presents a mainstream solution for hardware functional test automation.

A system built with similar tools and components as in [38] is in use in the Company for printed circuit board functional testing after component assembly and reflow soldering. Systems built for production testing could be adapted to system testing. Hardware probing interfaces and test and measurement equipment would be readily available, and software test automation scripts could be run in this environment as well. A key benefit of using commercial tools like LabView is the availability and support for test and measurement instrument drivers.

Test system description [39] by Keimling, Hansen, and Bilgic presents a complete solution for integration, regression and system testing of industrial process instrumentation devices. The presented system is also used for system hardware characterization and stress testing. In this system, a set of various tools have been used. Dedicated applications have been written in C++ for test sequencing and interfacing test equipment. The qt4 framework has been used for user interfaces of the applications. The system uses VISA, and other libraries from equipment manufacturers in the test instrument interfaces and the actual test sequences have been written in ECMAScript language. Also, test results analysis and reporting have been automated, and it is handled in Matlab and LaTeX tools. Authors explain that the whole system goal has been to make a modular system where new test instruments and interfaces to new systems under test can be added easily. Also, an automatic standardized report generation has been one of the main goals. Authors say developing a complete test automation system from ground-up with the various tools has taken ten man-months and speculate the effort has been reasonable compared to using commercial tools for the same system. However, the authors say the system is missing test sequence synchronization and test case management, which are to be developed next.

This example case presented in [39] shows that building the whole system from ground-up may be beneficial. Especially in cases where the embedded system under test has several proprietary interfaces, it may be easier to build a test automation system around some existing system built for interface and integration testing. However, the work needed to specify the test system software architecture and implement vari-

ous application programming and test equipment interfaces requires resources and time. The Company has such systems as well in place for example for device temperature compensation and calibration in production. These systems are for legacy products, and they are in maintenance mode. New systems are built on top of National Instruments LabView and TestStand to avoid building the test instrument interfaces and test sequencing from ground-up.

7.1.2 Ideas for Regression Testing

Paper [40] by Pekovic, Banika, Kuburovic, Zlokolica, and Vranic present a solution for simultaneous functionality verification of multiple digital TV set-top boxes. In the presented system it is possible to test multiple devices simultaneously with the same input and compare the outputs to an output of a reference device, which acts as the pass criteria in the tests. Authors say the parallel testing of multiple devices is needed in verification and stress tests which are repeated multiple times. Overall testing time can be reduced by running the tests parallel on many devices. This approach will give statistical data on different HW units, which is often necessary to evaluate hardware reliability. The authors do not disclose the details of the software tools used to implement the system. However, the overall architecture of the system is explained, and the key hardware component is an Ethernet-connected video and audio capturing device. Captured data from multiple video streams are stored on the test system and compared to the reference.

This idea of using a reference device as the pass criteria for a test and running the tests simultaneously for multiple devices could be used in regression testing as well. The Company develops patient monitor clinical software, which can be run on several different HW variants, but which graphical user interfaces share the same elements. Parallel regression testing of multiple hardware variants with new software could be used in the Company to speed up the regression testing, without a need to multiply test systems for each test bed. The image capturing together with screen scraping methods could be used to pick specific areas from the screens of reference device and devices under test. Automated test system could then run the same test scenarios on all devices and reference system simultaneously and observe the differences. Compared to setting up and running the tests individually for each HW variant, this kind of method could speed up the regression testing and make it possible with fewer test equipment. In order to enable this kind of testing, the automated tests system should have simula-

tors and emulators which can feed simultaneously the same input to several systems under test.

In paper [41] by Alper and Mehmet, a system intended for verification, manufacturing and regression testing of single beam echosounder system is presented. Echosounder is used in marine vessels to measure the depth of water. Authors explain that the primary motivation for building the system is to overcome problems with system tests performed on the ship with real hardware. Instead, performing system tests without real hardware platforms is seen as necessary in this case study. The testing system development included developing simulators and emulators for the system interfaces and real operating environment. Simulators and emulators were verified by comparing the operation of them to data gathered manually from the same tests in the real system and environment. Software development was made mainly with C# programming language. Authors mention that as a result of the automated test system development work, physical interfaces and functional features of the system can be tested. Compared to manual testing the automated testing system performs the same test set three times faster. Also, authors list other benefits of the system like being able to run tests continuously, repeating tests easily, testing in early stages, minimizing human error, and saving time compared to testing in a real environment. These benefits are apparent when considering the effort needed to assemble a system in real marine vessel environment or doing the same in a laboratory environment.

The system the Company is developing for its customers is modular. There are several different types of add-on modules and peripherals the user can add to the system, and which all have their dedicated hardware and software. Currently, the automated testing consists of multiple system set-up, which each have a predefined set of hardware modules and peripherals connected to the central unit. The automated tests are run for this predefined set. Changing to different set-up takes either extra time, or much extra equipment is needed to build another parallel set for testing. One of the key motivations explained in the paper [41] was to make testing easier and this was achieved mainly by simulating some interfaces of a real marine vessel and emulating real environment instead of always testing the full system in the real environment. Similar methods could be used in the Company as well. Emulators could be used to replace modules and peripherals and use them in regression testing. With emulators, there is always a question of how well they represent the real hardware and their embedded software. Building an accurate emulator can require more effort than running the tests with real hard-

ware as the emulators used in medical device development require verification and validation of their own. Due to this, verification and acceptance testing could be still executed with a full set-up or real hardware. However, while testing with real a complete system, data from the various interfaces could be recorded. With the recorded data emulators for regression testing could be developed. Simple emulators could be used to speed-up the regression testing. In regression testing typically a small set of tests are run, and results are not used as verification evidence. Development effort for the emulators could be limited in the first place to these specific tests, instead of developing an emulator implementing the piece of modular hardware and its software accurately.

7.1.3 Real-Time Requirements for System Stimulus and Parameter Measurement

In the Company the developed systems are for patient vital signs and other human physiological parameters monitoring in hospital and a clinical environment. Human physiological parameters change very slowly compared to the capabilities of the embedded system doing data acquisition and processing. The bandwidth of the signals is at maximum some kHz, and the system under test typically reacts to the signal changes in about one to two seconds and shows the values on the screen to the user or generates a visual or an audible alarm. Due to this, there are no strict requirements for test automation system real-time data processing. If hardware parameters were monitored during the system testing, it would be adequate to store information about the test case in progress and time stamp with a resolution of one second together with the measured value from hardware.

Paper [42] by Antonelli, Bucci, Ciancetta, and Fiorucci presents a system where the synchronization of stimulus and measurements has had strict requirements. In the paper, an automated test system for avionics electro-mechanical actuators is presented. In this case study, the system is built for testing the actuators by emulating different working conditions the system might experience in a real-life system. The motivation for building the system has been to gather more information from the experimental testing to better understand various failure modes the system under test may have. The system has been built mainly with commercial off the shelf data acquisition modules and test equipment. The test system software has been developed with National Instruments LabWindows CVI. Authors explain this selection by being able to use the same toolchain for generating graphical user interfaces and allowing to use multi-

threading, event-based programming. Also, the same environment is used to implement both software and hardware interrupts and interfacing with data acquisition and external devices with the readymade libraries. In the paper, the methods used to synchronize the system stimuli and gathering measurement data from test equipment is explained in great details. As a result, the authors have been able to build a system which can gather data with required accuracy in the time domain and which allows analyzing the system under test behavior in various load conditions.

The methods presented in [42] could be used in the Company if some real-time events in the system hardware need to be captured. Synchronizing multiple measurements could be needed for example when listening to some data interface, using captured data pattern as a trigger and which causes the recording of several analog values or waveforms after it.

7.1.4 Inrush Current Monitoring to Reveal Problems in System

Paper [43] by Sun, Chen, Wang, and Wang discuss production testing of a combine harvester electronics and present a test system developed for it. Authors mentioned that 30% of the field failures in combine harvesters could be traced to the electrical system and comprehensive testing of it in production is a way to improve the product quality. The presented test system can check all electrical units and functionalities of the system, and it uses supply power current measurement as an indication of a working unit. Additional measurements have been added to the system to analyze the battery current in the time domain while the combine harvester engine is being started. The current waveform is used to calculate startup time and together with measured voltage drop are used as indications of a functional system and a correct assembly. In the system, the power supply current is measured with current clamps. Also, voltage, rotation speed, and communication bus are monitored by the system. Software tools used for system development are not disclosed in the paper. Test system stores data of each tested unit and has features of presenting the results over time in table or graph format.

A similar method of inrush or idle current consumption is used in the Company on the printed circuit board assembly functional testers. The current measurement is used usually as the first test of the board after reflow soldering to check there are no short circuits on the board and it is safe to proceed with releasing the system reset and turn-

ing on other power supplies. Making the power supply start-up sequence safe for the system and designing the different power domains to minimize inrush current is a standard step in embedded system hardware design. Functional testing and verification are typically done in the HW block unit testing and characterization. However, due to the nature of the systems the Company design and manufacture, it is difficult to test all possible combinations of the modular system and how the user may connect or disconnect modules. In addition, it is not possible to make the connection and disconnection of the modules in all possible system states to verify that the system keeps running normally and produces the expected response. If the connection and disconnection of the system modules could be automated, a full system repeated random sampling techniques, or Monte Carlo techniques could be used to have better testing coverage for module connection and disconnection. In this kind of connection and disconnection testing, measurement of module input power could be added to obtain numerical results of the inrush current in various system states and combinations. Executing a full coverage connection and disconnection testing would be possible only with test automation. Manually going through all possible combinations would not be feasible due to the very long execution time.

7.1.5 Literature Review Conclusions

The literature review and study of the test systems showed the open source and Python script-based Robot framework, already in use in the Company for system testing automation, can be expanded with test and measurement equipment and could also handle system level hardware testing. This observation can be used as a basis for recommendations for automated testing system improvements. Prototyping for example with an oscilloscope and measuring patient monitor audio or visual indicators output could be the first step. In addition, the literature review gave two ideas which could be applied in the Company as well. First, the method presented in [40] could be used to monitor the patient monitor display output and compare the system under test to a reference device. Automatic image recognition and character recognition could be used to evaluate the clinical application user interface output against the reference device and detect the differences automatically. Another applicable method could be the system and module supply power monitoring while running test cases and use scenarios. This method is inspired by the method presented in [43]. The testing system could in the first phase gather data on system power consumption and interface supply current changes in different system states. Later, this database could be used to observe sys-

tem software changes effect on system and module power consumption and for observing if there is a correlation between defects and power consumption changes.

7.2 Interviews

Interviews were carried out during 2018. A semi-structured qualitative interview of electronics and system design engineers in the Company was organized to gather data. The questions and interview outline are described in Appendix 1 – Interview Guide. The questions and topics discussed with respondents were:

- What kind of hardware related defects has the expert faced?
- How was the defect discovered?
- Could the defect have been discovered in automated system testing?
- What kind of test equipment and signaling to test system would have been required to automate the test?
- Based on the expert's experience, what kind of measurement techniques should be considered to improve automated testing?

Appendix 3 has summaries from five interviews. These interviews took approximately 35 – 50 minutes each. Respondents of these interviews had 14 – 44 years and on average 27 years of experience in electronics design. Respondents have been working with industrial, consumer and medical electronics. Two of the respondents had bachelor's, and three had a master's degree in electronics. Respondents described a total of 14 different cases, where a defect in system software or hardware component had caused a failure in the system.

The data from the interview was analyzed and categorized in the following five themes:

- Failure mode
- Failure detection phase
- Detectability
- Test automation in use?
- Improvement ideas and opinions

From the case examples the respondents had experienced, the failure mode is the key data point for this study. It tells how the system had failed and what was the defect root

cause. This information can be used to plan improvements which have the most impact on finding defects in automated system testing.

Failure detection phase and detectability were analyzed from the case examples and respondent answers to have an idea in which phase of the product development the defect was detected and have the respondent's opinion on the defect detectability. This data point helps to conclude how often a defect is undetected in actual product development phase.

The question of if any test automation was in place helps to understand better how the defects respondents described had been detected. The fifth central theme to categorize the data was the respondent's improvement ideas and opinions on system test automation. These ideas can be used directly as input to system test automation future improvements.

Under each main theme, subthemes were created to group the data from respondent's case examples and ideas. From each subtheme, the number of individual cases and opinions or ideas were counted. The subtheme and count allow finding patterns on similar failure modes, ideas, recommendations and conclusions from the respondent's answers. Table 2 (p. 45) shows the created themes and the subthemes counted from them for the data analysis.

The data shows some clear patterns. Majority of the defects the respondents described were either due to HW component or electronics circuit defects or related to power supplies. Also, the majority of the defects were found in system testing phase, but in most of the cases, there was no system test automation in place. In all cases presented, the respondents thought that the defect could have been detected in automated testing if the test system would have certain testing capabilities in place.

Each respondent thought that test automation is useful. However, test automation should not concentrate only on system testing, but the unit testing coverage should be improved to find the defects before system integration and system testing is started. Majority of the respondent's improvement ideas for system test automation were related to test automation of unit or module connection and disconnection and power supply testing.

Table 2. Interview data analysis and grouping.

Main Theme	Subtheme	Respondent					Count
		1	2	3	4	5	
Failure mode	HW component or electronics circuit functional defect			2	1	2	**5**
	Power integrity defect			1	1	1	**3**
	Device fails to start after a power cycle.	1		1			**2**
	Software functional defect: -Connection or disconnection of the device caused communications failure -Wrong behavior of the software		1	1			**2**
	Timing failure (real-time scheduling issue)		1				**1**
	Operating environment humidity impact		1				**1**
Detection phase	Defect detected in unit testing	1		1			**2**
	Defect detected in system testing		2	4	1	2	**9**
	Defect detected in production		1				**1**
	Defect detected by the customer				1	1	**2**
Detectability	The customer would have detected the defect	1	3	5	2	3	**14**
	The defect could have been detected in automated testing	1	3	5	2	3	**14**
Test automa-tion	Unit test automation system in place	1					**1**
	No unit test automation		3	5	2	3	**13**
	System test automation in place		1				**1**
	No system test automation	1	2	5	2	3	**13**
Improvement ideas and opinions	Test automation usefulness	1	1	1	1	1	**5**
	Improve unit testing coverage	1	1	1	1	1	**5**
	Power cycling (with variable timing)		1	1	1	1	**4**
	Unit connect / disconnect testing automation	1	1	1			**3**
	Breakout board or test connectors to make HW unit testing and system testing easier.	1		1			**2**
	Supply power fluctuation injection		1			1	**2**
	Communications fault injection		1				**1**
	UI element and display information automated testing		1				**1**

7.2.1 Interview Conclusions

The interviews showed two clear themes which can be used as a basis for recommendations for automated testing system improvements. These themes were related to the failure mode and defect detection phase.

The first observation was, that out of 14 cases presented by the respondents, in 12 cases the defect was detected either in system testing, in production or by the customer. However, in each case, the respondent thought the defect could have been detected earlier in unit testing. All respondents also presented an idea of improving the unit testing and were of opinion automated testing is useful. Also, approximately in 10 out of 14 cases, the defect root cause was a component or unit design error or functional failure in the unit. These observations suggest the system testing automation improvements, to better test the hardware should be considered, but at the same time reviewing the electronics design process and improving the coverage of testing made earlier in the project could also have a significant impact on finding defects from the design before the system integration and testing.

Another observation from the cases presented by the interview respondents was the number of failures caused by power integrity, device power cycling or module connection and disconnection. Approximately in 50% of the cases, the defect root cause was related to these basic use scenarios. Failures related to inrush current or device functionality at minimum or maximum supply voltage should be possible to detect early in the project. However, the effect of various combinations of use scenarios, connected peripherals, and system software services available at the time of testing may lead to a situation that testing is not possible until at the time of system integration and system testing. Because of this, powering the system on and off with various configurations of peripherals and modules, or hot plugging them could be a useful feature in automated system testing for finding defects. Manual testing, in this case, is not feasible. The automated test system infrastructure should be built to enable the power cycling and connection and disconnection tests.

7.3 Peer Review Input

Verification of the intermediate study output was arranged as peer review among the experts interviewed during the study, and among experts developing the automated testing system.

Feedback from the peer review confirmed the assumption that automated system testing could be used to find hardware defects related to system power integrity, interrupt and code execution timing variability, and peripheral connection and disconnection. However, to implement test cases stressing the system hardware, the automated system test development team needs instructions and guidance documents from the HW development team. Also, there is a need for purpose-built hardware to make the necessary connections between the system under test and test and measurement equipment. In other words, the HW engineers and automated system testing developers should work closely together when defining test cases.

Important topic highlighted by the automated testing system developers was the fact that the automated tests act, at the moment, only on the edges of the system under test. This means that for example, the test system is not capable of measuring delays or executions time inside the system. Timing can be measured from system input to output signals. If a test case requires measuring a specific timing inside the system, it would require new testing methods. One idea for measuring timings inside the system could be a dedicated test and measurement equipment connected to systems internal signals and use them as triggers or flags about the event start, stop or progress conditions.

In addition, the ease of building the test set-up was found to be important. If new test and measurement equipment are introduced, the preferred way would be a single connector in the device under test where any controlling and measuring devices can be attached. Keeping the system cabling simple and clutter free is essential for error-free, fast, and efficient test set-up building. Wish from the automated system testing development team was to have purpose-built break-out boxes and multiplexers to enable connecting and disconnecting peripherals, their power supplies and swapping different types of devices between interfaces. Ideally, the break-out boxes should be controllable through the Ethernet interface to enable easy instrumentation and connectivity to the computer controlling the testing sequence.

8 Recommendations

Below are the recommended actions to consider, when planning improvements to the automated system test infrastructure in the Company. These recommendations are based on the conclusions from the literature review and analysis of the data from the interviews. The recommendations are organized in order of priority for improving the overall capability to reveal hardware defects.

1. Add equipment in system test automation infrastructure to allow cycling power supplies, for generating controlled shutdown and power-up sequences and disconnecting and re-connecting system modules and peripherals supply power or communications. These functionalities would allow fully automating the testing of these use scenarios and emulating the user interaction with the system.
2. Add an interface and methods to the test system for adding measurement equipment. Mixed signal oscilloscope with the capability of measuring analog and digital signals could be the first step. The output from the oscilloscope could be used in system test automation to confirm some functionality under test has been performing as intended, and all the hardware and software interfaces function correctly with new software releases.
3. Add capabilities to measure system and peripheral modules power consumption. Database showing power consumption figures in various tests and use scenarios can be used later to help to find the root cause to system defects and observing system software changes effect on power consumption.
4. Review the electronics design process and possibilities to improve the unit and block level testing. Also, unit and block level testing automation should be considered. Standard interfaces functionalities which do not depend on system software could be tested already at the unit level.
5. Consider adding capabilities to the automated testing system for display image capturing and adding image and character recognition. Automating the inspection of user interface elements could improve the automated testing coverage and reduce manual testing further.

Table 3 (p. 49) shows an effort and impact analysis for the recommendations. Effort score of 1 means least effort is needed to implement the recommendation. Impact score of 1 means the highest positive impact on improving the overall capability to reveal hardware defects. The total score is calculated by multiplying the effort and impact and can have values from 1 to 25. The total score is used to set the recommendations in the order of priority, and where the lowest total score indicates the highest priority or the first improvement to consider to be implemented. Figure 8 (p. 50) shows the effort and impact analysis total score as a matrix and places the recommendations 1 – 5 to the matrix.

Table 3. Effort and impact analysis on the recommendations.

Recommendation	Effort score	Impact score	Total score	Discussion
1. Cycling power supplies and disconnecting and reconnecting system modules and peripherals.	2	1	2	Building break-out boards and control mechanisms for connecting and disconnecting peripherals and cycling the system under test power is easy to implement. Impact of this kind of features would be high, as the interviews showed that many times an observed defect was due to malfunctioning HW during power cycle or peripheral device connection or disconnection.
2. Interface and methods to the test system enabling HW test and measurement equipment.	2	3	6	Preliminary study and start learning how to add test and measurement equipment to the existing system could be started simply by adding for example oscilloscope to the system. As a first step, an oscilloscope could be used to check simple user interface elements like LED's or speaker functionality. The effort to do this is low and requires some learning. The impact is moderate, as the number of test cases which would need this kind of hardware monitoring is low.
3. Equipment to measure system and peripheral modules power consumption.	3	3	9	In order to measure the power consumption of system modules, new test and measurement equipment would be needed for the testing system. The effort of developing the interface and breakout boards and writing methods for them is medium. Impact of this new feature of the system is medium. Measurement data should be gathered for some time, and it should be analyzed to be able to draw conclusions about the power consumption in different system use scenarios. After analysis, the measurement could be used in real test cases as pass and fail criteria or as a warning about changes in system behavior.
4. HW unit testing and electronics development process improvement.	2	5	10	Reviewing the electronics design process and building in mechanisms for hardware unit testing is a long-term development project. This should be considered as an evolution of the Company culture and discipline towards the better total quality of the electronics design. The effort for this activity is moderate, but the impact on finding HW defects early in the project is high.
5. Equipment for display image capturing and adding image and character recognition to the system.	5	3	15	Analyzing the content of the display could be used to automate various user interface tests and to gather feedback about the system behavior in various test cases. However, the impact on capabilities on finding HW related defects is medium or low, as the display itself does not usually reveal defects on other parts of the HW. The effort to implement this kind of features is high and would require new test equipment and learning how to use image recognition methods efficiently.

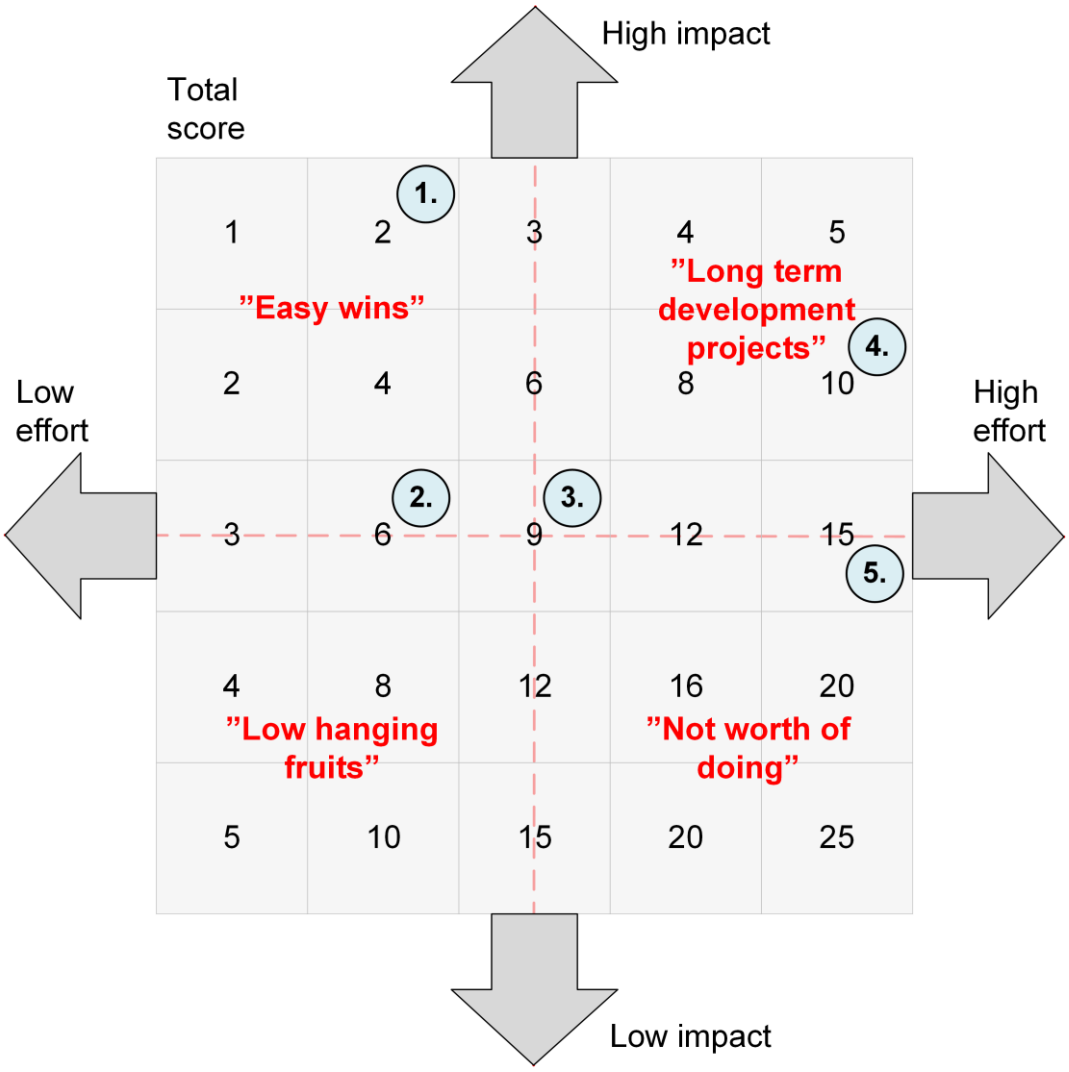


Figure 8. Effort and impact analysis matrix and total score values

9 Conclusions

In this thesis report, embedded system hardware development methods and integration of HW testing to automated system testing are discussed. To find out what kind of tests and testing equipment should be added to the automated system testing infrastructure, this study used literature review and expert interviews as methods. Idea's from case examples from literature and experiences from previous product projects were used to draw conclusions of the current state and possible improvement areas. As a result of the study, the literature review and expert interview conclusions were used as basis for a prioritized list of recommendations for improving the automated system testing and electronics design process.

The literature review gave some ideas which can be developed further to come out with new testing methods. Especially some of the recommended methods could be used to monitor the system under testing better in regression testing. In addition to ideas for new testing methods, the literature review gave confidence that the existing automated system testing infrastructure in use can be developed further to handle system level hardware testing as well.

The expert interviews revealed some weakness in the ways the embedded system electronics are being developed in general. Thematic analysis of the interviews showed that special attention on the electronics design process should be paid to detect the hardware defects and design errors early in the project. Another theme that came up in the interviews was that many of the problems had been related to failures caused by power integrity, device power cycling or module connection and disconnection. The electronics design process could be built such that it enables finding most of the defects already at hardware unit testing. Unfortunately, hardware unit testing cannot reveal all defects. Some system functionalities require that the whole system is tested, and this can be made in the automated system testing.

Validation of the recommendations was made as a peer review. The peer review confirmed the recommendations developed could have a positive impact on the ways the Company develop electronics and uses the automated system testing infrastructure for improving total product quality.

In addition to peer review, some of the recommendations could have been tested in the real automated system testing infrastructure, but that was not in the scope of the study. Even if no real hardware or detailed testing methods were developed to improve the testing system, some of the work has been started to realize part of the recommendations. While writing this thesis report some of the recommendations were drafted already when a small automated system testing infrastructure improvement project was started. Ideas from this study were used to build one improvement to the automated system testing. The author participated in defining a new device, which can disconnect and reconnect the clinical parameter modules power supply and data communications and measure the power consumption of each module. The new device, when integrated into the automated test system, allows experimenting with random or sequenced module connection and disconnection and measuring the steady-state power consumption and inrush current. This data can be used to develop the system hardware further to handle varying power consumption scenarios better and experiment with interrupts and events generated by module connection and disconnection.

While writing the thesis report, the methods and materials used in this study raised some additional ideas on distributing knowledge and lessons learned within the Company engineering community. The conference proceedings and research papers published in the field of test automation can provide lots of information and new ideas for improving test automation. The literature review made in this thesis was limited to papers describing full test automation systems architecture in general level. However, the papers describing a specific method for a single functionality or interface could have valuable information which could be utilized in the Company as well. If a specific problem is identified, a new literature review could be arranged to find relevant information from the conference proceedings and research papers. Another way of distributing information within the Company could be to gather lessons learned from observed defects more widely. The interviews arranged in this thesis study could be continued and a workshop presenting the cases and ideas raised by the lessons learned could be a useful way to gather information for improving the test automation processes and practices.

References

[1] Python Software Foundation, "Python Software Foundation, PyVISA project," [Online]. Available: https://pypi.org/project/PyVISA/. [Accessed 7 December 2018].

[2] SCPI Consortium, "1999 SCPI Syntax & Style," IVI Foundation, 1999. [Online]. Available: http://www.ivifoundation.org/docs/scpi-99.pdf. [Accessed 20 January 2019].

[3] IVI Foundation, "IVI Specifications," [Online]. Available: http://www.ivifoundation.org/specifications/default.aspx. [Accessed 7 December 2018].

[4] E. P. a. Council, "Regulation (EU) 2017/745 of the European Parliament and of the Council of 5 April 2017 on medical devices, amending Directive 2001/83/EC, Regulation (EC) No 178/2002 and Regulation (EC) No 1223/2009 and repealing Council Directives 90/385/EEC and 93/42/EE," *Official Journal of the European Union,* vol. 60, no. L 117, pp. 1-175, 5 May 2017.

[5] I. T. 210, "EN ISO 13485:2016 - Medical devices. Quality management systems. Requirements for regulatory purposes (ISO 13485:2016)," CEN-CENELEC, Brussels, 2016.

[6] C.-C. T. 3, "EN ISO 14971:2012 - Medical devices – Application of risk management to medical devices (ISO 14971:2007, Corrected version 2007-10-01)," CEN-CENELEC, Brussels, 2012.

[7] I. T. 62, "EN 60601-1 - Medical electrical equipment Part 1: General requirements for basic safety and essential performance (IEC 60601-1:2005)," CENELEC, Brussels, 2007.

[8] I. S. 62A, "EN 62366-1:2015 - MEDICAL DEVICES. PART 1: APPLICATION OF USABILITY ENGINEERING TO MEDICAL DEVICES (IEC 62366-1:2015)," CENELEC, Brussels, 2015.

[9] I. T. 210 and I. S. 62A, "EN 62304:2006 - Medical device software - Software life-cycle processes (IEC 62304:2006)," CENELEC, Brussels, 2006.

[10] I. S. 62D, "IEC 80601-2-49:2018 - Medical electrical equipment - Part 2-49: Particular requirements for the basic safety and essential performance of multifunction patient monitoring equipment," International Electrotechnical Commission (IEC), Geneva, 2018.

[11] A. Engel, Verification, Validation, and Testing of Engineered Systems, Hoboken, New Jersey: John Wiley and Sons Ltd, 2010, p. 715.

[12] D. A. Vogel, Medical Device Software Verification, Validation, and Compliance, Norwood , Massachusetts: Artech House Publishers, 2010, p. 428.

[13] P. Tripathy and K. Naik, Software Testing and Quality Assurance: Theory and Practice, Hoboken, New Jersey: John Wiley & Sons, Inc, 2008, p. 616.

[14] robotframework.org, "Robot Framework - Introduction," robotframework.org, [Online]. Available: http://robotframework.org/#introduction. [Accessed 5 May 2018].

[15] P. Laukkanen, "Data-Driven and Keyword-Driven Test," Helsinki University of Technology, Espoo, 2006.

[16] S. Bisht, Robot Framework Test Automation, Birmingham: Packt Publishing Ltd, 2013, p. 112.

[17] D. Sale, Testing Python - Applying Unit Testing, TDD, BDD and Acceptance Testing, Chichester: John Wiley & Sons, Incorporated, 2014, p. 243.

[18] National Instruments Corporation, "Fundamentals of Building a Test System," [Online]. Available: http://ftp.ni.com/evaluation/coretest/Fundamentals_of_Building_a_Test_System_CompleteGuide.pdf. [Accessed 27 December 2018].

[19] P. Arpaia, E. D. Matteis and V. Inglese, "Software for measurement automation: A review of the state of the art," *Measurment - Journal of the International Measurement Confederation (IMEKO),* vol. 66, no. April 2015, pp. 10-25, 2015.

[20] National Instruments Corporation, "3rd Party Instrument Drivers," National Instruments, [Online]. Available: http://www.ni.com/downloads/instrument-drivers/. [Accessed 27 December 2018].

[21] P. Authors, "PyVISA Documentation, Release 1.10.0.dev0," 02 December 2018. [Online]. Available: http://media.readthedocs.org/pdf/pyvisa/latest/pyvisa.pdf. [Accessed 19 January 2019].

[22] P. M. Bormpantonakis, D. I. Stratakis, G. N. Mastorakis, C. N. Skeberis, C. X. Mavromoustakis and P. V. Bechet, "Exposure EMF measurements with spectrum analyzers using free and open source software," in *2016 International Conference on Telecommunications and Multimedia (TEMU)*, Heraklion, Greece, 2016.

[23] E. Apostolaki-Iosifidou, N. Ramachandran and L. Dong, "Server Power Supply Test Automation Approach," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Houston, TX, USA, 2018.

[24] M. Kerttula, Virtual Design. A Framework for the Development of Personal Electronic Products, Espoo: VTT Technical Research Centre of Finland Ltd, 2006, p. 218.

[25] T. Punkka, "Agile Hardware and Co-Design," in *Embedded Systems Conference*, Boston, 2012.

[26] G. L. B. Lima, G. A. L. Ferreira, O. Saotome, A. M. d. Cunha and L. A. V. Dias, "Hardware Development: Agile and Co-Design," in *12th International Conference on Information Technology - New Generations*, Las Vegas, 2015.

[27] G. L. B. Lima and O. Saotome, "CoH-Agile: Proposal of methodology for development HW/SW embedded," *International Journal of Enhanced Research in Science Technology & Engineering,* vol. 4, no. 5, pp. 91-94, 2015.

[28] M. White and J. B. Bernstein, "Microelectronics Reliability: Physics-of-Failure Based Modeling and Lifetime Evaluation," Jet Propulsion Laboratory, Pasadena, 2008.

[29] K. A. Gray and J. J. Paschkewitz, Next Generation HALT and HASS : Robust Design of Electronics and Systems, Chichester, West Sussex: John Wiley & Sons, Ltd, 2016.

[30] J. T. DiBene, Fundamentals of Power Integrity for Computer Platforms and Systems, Hoboken, New Jersey: John Wiley & Sons Inc., 2014.

[31] Y. Kim, K. Kim, J. Cho, J. Kim, K. Kang, T. Yang, Y. Ra and W. Paik, "Power Distribution Network Design and Optimization based on Frequency Dependent Target Impedance," in *IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, Seoul, 2015.

[32] S. A. Chickamenahalli, M. Suryakumar, E. Stanford and K. Merley, "Effect of target impedance and control loop design on VRM Stability," in *APEC. Seventeenth Annual IEEE Applied Power Electronics Conference and Exposition*, Dallas, TX, USA, 2002.

[33] J. Kim, S. Wu, H. Wang, Y. Takita, H. Takeuchi, K. Araki, G. Feng and J. Fan, "Improved Target Impedance and IC Transient," in *2010 IEEE International Symposium on Electromagnetic Compatibility*, Fort Lauderdale, FL, USA, 2010.

[34] P. Regnier, G. Lima and L. Barreto, "Evaluation of Interrupt Handling Timeliness in Real-Time Linux Operating Systems," *ACM SIGOPS Operating Systems Review,* vol. 42, no. 6, pp. 52-63, October 2008.

[35] M. Dai and Y. Ishikawa, "Delayed Interrupt Processing Technique for Reducing Latency of Timer Interrupt in Embedded Linux," in *International Conference on Computational Science and Engineering*, Vancouver, 2009.

[36] F. Scheler, W. Hofer, B. Oechslein, R. Pfister, W. Schröder-Preikschat and D. Lohmann, "Parallel, hardware-supported interrupt handling in an event-triggered real-time operating system," in *2009 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES 2009*, Grenoble, France, 2009.

[37] C. Rochange, S. Uhrig and P. Sainrat, Time-Predictable Architectures, London: ISTE Ltd, 2014.

[38] V. D. Ćatić, N. M. Lukić, I. M. Salom, V. P. Ristić, M. M. Kabović and N. M. Nenadić, "An automated environment for hardware testing using PXI instrumentation and LabVIEW software," in *2016 24th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, 2016.

[39] R. Keimling, C. Hansen and A. Bilgic, "A system for automated testing in development of measuring devices for industrial process instrumentation," in *JAMAICA 2013, Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to testing Automation*, Lugano, Switzerland, 2013.

[40] V. Pekovic, D. Banika, D. Kuburovic, V. Zlokolica and N. Vranic, "Simultaneous functionality verification system of multiple Set-Top Boxes," in *2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems*, Novi Sad, Serbia, 2012.

[41] A. Avcioglu and M. Demirer, "Implementation of System Testing Automatization on Computer Aided Systems for Hardware and Software," in *2015 IEEE AUTOTESTCON*, National Harbor, MD, USA, 2015.

[42] M. G. Antonelli, G. Bucci, F. Ciancetta and E. Fiorucci, "Automatic test equipment for avionics Electro-Mechanical Actuators (EMAs)," *Measurement - Journal of the International Measurement Confederation (IMEKO),* vol. 57, no. 2014, p. 71–84, November 2014.

[43] D. Sun, D. Chen, S. Wang and X. Wang, "Development on electrical system performance test stand for combine harvester," in *6th IFAC Conference on Bio-Robotics BIOROBOTICS 2018*, Beijing, China, 2018.

[44] T. Ung, T.-C. Nguyen, L. Vuu, B. Barrus, M. Reimann, S. Archibald, S. Rawlings and J. Tuft, "Radio Frequency Test Set: An Ethernet-Based RF ATE System Designed to Test a Minuteman Telemetry Wafer," in *2016 IEEE AUTOTESTCON*, Anaheim, CA, USA, 2016.

[45] X. Yang, Y. Lin, F. Gao, G. Wang and Y. Zhang, "Automated Test System Design of Body Control Module," in *2014 International Conference on Information Science, Electronics and Electrical Engineering*, Sapporo, Japan, 2014.

Appendix 1 – Interview Guide

Introduction

As part of this master's thesis research, electronics design engineers and system design engineers are interviewed for collecting their knowledge, experiences, and opinions on the automated system testing.

The interview used in this study is a semi-structured qualitative interview on topics around the following key questions:

- What kinds of failures are not discovered until at system testing?
- Which hardware parameters are important to monitor in automated testing of embedded system?

Interview procedure

Each interview is organized as a face-to-face interview with a single respondent. One interview takes approximately 1 hour.

The research topic and the current state of the automated system testing are introduced to the respondent briefly. Next, the respondent is explained how the collected information is used and stored. After the introduction, the respondent own experiences, knowledge and opinions are collected.

At the end of the interview the respondent is asked to share some data about his/her working career length and from which fields of electronics industry he/she has experience. This data is used to explain statistics on the whole interview respondent population.

The interview is recorded on the tape and transcript of the interview is written. The interview transcript is written in a way that persons, companies or products cannot be identified. Respondent is asked to review the transcript and give feedback and any additions he/she would like to highlight. After approval from the respondent, the materi-

al from the interview is added as input data to the study. Interview recordings are not stored; they are used only to enable interview transcription and are deleted after it.

Study background and introduction to the respondent

Embedded system electronics design usually follows a company-specific product development process. The development process may include phases like product concept study, specification, detailed design, verification, and production ramp-up. The phases may take advantage of iterative and incremental design methods. The phases may overlap, and development work is done in small steps or at the functional block level.

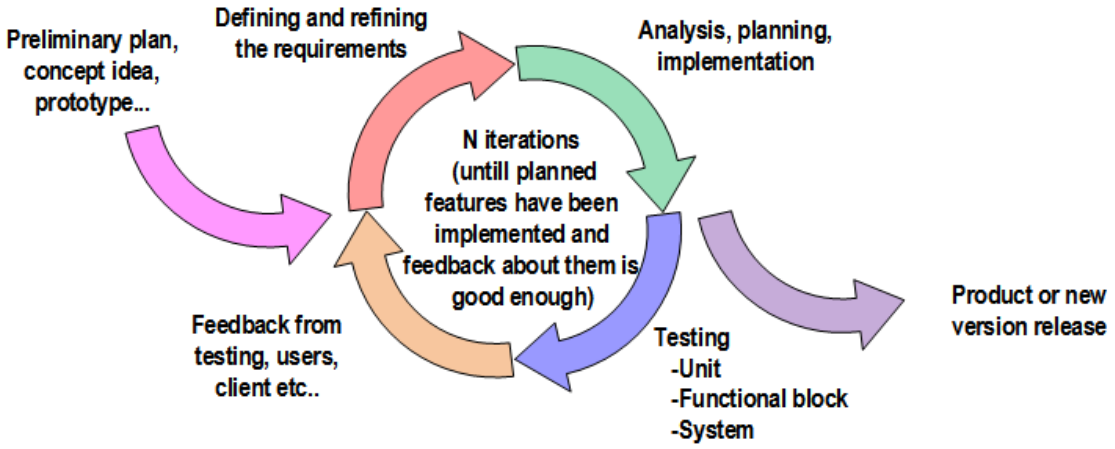


Figure 9. The idea of the iterative and incremental development process for product development. Figure modified by Antti Teronen from the original file in [A1/1].

Often the design practice is, that the electronics functional block specification, detail design, and verification include testing which covers expected and extreme conditions for the functional block and its interfaces. Testing may include for example testing of functionality over wide operating temperature and voltage ranges. Also, protections against foreseeable misuse and failures are typically tested at the functional block level.

The functional block level testing may reach high testing coverage for the block, but testing is usually done in static conditions. When the whole system is put together, the number of combinations of different events across the system can be in practice infinite and new problems and failures, which were not seen at block level tests, may be ob-

served. As an example, the variability of event timings caused by interrupt handling latencies and software thread execution order may introduce a condition in the system which was not expected. Another example is the system level power integrity and unpredicted load and line transients and noise coupling, which may cause problems or failures in the functional blocks.

Due to the reasons and examples mentioned above the system level testing is an integral part of the system verification and cannot be left away from product development. To increase the system level testing coverage and to speed up the testing, the system level testing is often automated. Automation may include for example a robot simulating the user and software scripts creating varying and challenging use scenarios. Testing may include also simulated failures for observing the system recovery from those.

This Master Thesis research concentrates on the embedded system hardware failure mechanisms and monitoring hardware during automated system testing. The research goal is to gather information about what kind of failures have been observed in the past, what has been the root cause, and would it been possible to detect such failure in the automated testing.

Information confidentiality

Example cases from real life are critical and useful for this research, as they make it easy to understand the problem the respondent has faced and how it may have been solved. However, to keep the information anonymous, the respondents are asked to keep their answers generic and not to mention any specific company or product they have been developing. The transcript of the interview will mention only the field of the electronics industry, like medical devices, industrial automation or mobile communications, but the specific company or product is not going to be specified.

The respondent name is not stored with the transcript. The recording of the interview is deleted after the respondent has reviewed the interview transcript and approved or denied its use for the master's thesis research.

Interview questions

The interviews should follow the practice highlighted by Robert S. Weiss [A1/2 p.48].

> "Permitting the respondent to talk about what the respondent wants to talk about, so long as it is anywhere near the topic of the study, will always produce better data than plodding adherence to the guide."

The questions are shown in Table 4 below forms a framework for the interview.

Table 4. Interview questions

Question	Follow-up questions to consider
What kind of hardware related defects have you faced during your career?	How was the defect discovered? Were the defects related to electronics, software of both? Have any of the defects been random or difficult to reproduce?
Was there automated system testing in place?	Could the defect have been discovered in automated system testing?
What kind of test equipment would have been required to automate the test?	How could the test system define when to trigger a hardware measurement? What kind of log files should the test system generate?
Based on your experience, what kind of measurement techniques should be considered to improve automated testing?	Ideas to consider for developing automated testing further? Ideas to consider for improving the test system and tools reusability?
How many years of experience you have from electronics of system design?	
With what kind of products have you been working with?	

References

A1/1 User:Aflafla1, File:Iterative development model.svg, Available at: https://commons.wikimedia.org/w/index.php?title=File:Iterative_development_model.svg&oldid=224424728, Accessed: 2018 February 10.

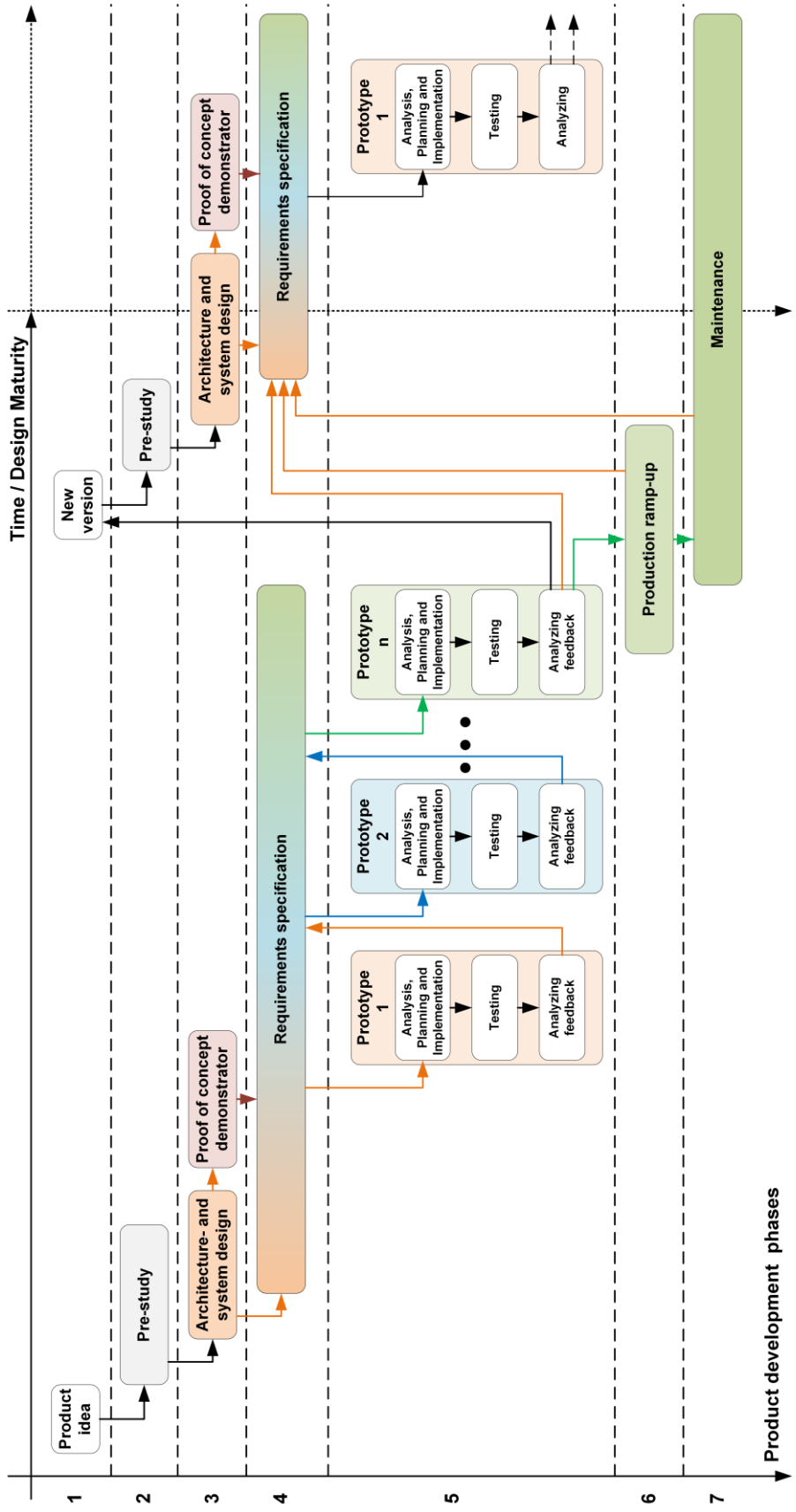
A1/2 Robert S. Weiss. Learning from strangers: The art and method of qualitative interview studies. New York, The Free Press, 1994, ISBN 0-02-934625-8, 246 pages

Appendix 2 – Embedded System Product Development Process Example

Figure 1 of this appendix is showing a simplified product development process which could be utilized in embedded system design.

In the example, there are 7 phases. The phases and activities during them are:

1. **Product idea**
   When a new product development project is started, the functions and features may not all be clear and specified. A new product idea or list of changes to the current product may be available as a starting point for the project.
2. **Pre-study**
   The pre-study phase is for gathering a more detailed list of features and functions the new product shall have. A list of items which are critical to quality (CTQ) can also be created.
3. **Proof-of-Concept**
   In the proof-of-concept phase, the architecture and system design are started, and a rapid working prototype or demonstrator is built. This phase is usually the final checkpoint for a decision to start the actual product development project.
4. **System Requirement Specification**
   The proof-of-concept phase is the key input to the system and product requirements. Also, system interfaces, customer and regulatory requirements are gathered. The baseline of the requirements is created as a starting point for the product design. However, requirements are refined throughout the whole project.
5. **Product design**
   In the design phase, the actual product is being developed. Development is made with iterations which add and refine features and functionalities to the product. After each iteration, the testing and verification results and feedback are analyzed, and input to the new iteration is gathered. Product requirements are also continuously fine-tuned based on the feedback. Hardware design cycles are mainly tied to the prototypes. System software can have its own shorter iterations. When the product is mature enough, the final iteration is ready for release for production.
6. **Production Ramp-up**
   After a successful system testing the product is released to production. Items identified as improvement opportunities are gathered and used as input for new product increment.
7. **Maintenance**
   System hardware and software are updated if necessary, and requirements for new product increment are gathered from production, service and customer feedback.



A2 - Figure 1. Example of product development process utilizing iterative and incremental development method with multiple prototype rounds.

Appendix 3 – Interview Transcript Summaries

Below is a summary of each of the five interview transcripts. As explained in the interview guide, the personal information of the respondents and detailed information on the products or companies has been removed from the summaries to keep the information anonymous.

Respondent 1

What kind of hardware related defects have you faced during your career?

Respondent gave an example from a project in consumer electronics. A power management IC did not start every time after a power cycle. The start-up failure occurred randomly in room temperature and was more frequent in cold temperature.

Could the defect have been discovered in automated system testing?

The defect in the product was discovered before the product was integrated into the final product and its system testing. The defect would have been discovered by the customer when the final product, targeted to consumers, would have been tested.

What kind of test equipment would have been required to automate the test?

Respondent explained there was a complex automated testing system in use for the component, which was used for debugging the observed defect.

The test system was able to connect measurement devices to the product under test through switching matrix and adapter boards. The adapter boards allowed fast testing of multiple devices. The adapters and the product under test were fitted in a thermal chamber to allow taking the ambient temperature as one variable in the tests.

The test system in use was also modeled into the simulation environment. This enabled transferring simulation cases to test cases in the actual physical tester.

Based on your experience, what kind of measurement techniques should be considered to improve automated testing?

Respondent presented an idea of a mini-board for power management IC input and output testing. The end product could have a standardized method for connecting a small printed circuit board to power management IC input and output for load transient, input under and over voltage, load regulation and output short circuit tests.

Also, the respondent experience has been that any method for making it easier to break a connection and making the test device connected to the electronics board would make it easier to test and debug the electronics board.

According to the respondent, the test automation for hardware has benefits like:

- Same tests can be repeated for each HW version to verify changes
- Testing over environmental conditions is easier as test cases can be run multiple times automatically.

Respondent 2

What kind of hardware related defects have you faced during your career?

Respondent has extensive experience on analogue electronics.

One topic the respondent wanted to be explicitly highlighted is that the effects of humidity should be thoroughly tested. Respondent gave an example of analog front-end amplifiers. In most of the cases, the front-end gain and the input impedance of the analog front-end is very high. Very low-level leakage currents to, from and between the input pins may cause false signals at the amplifier output. In a case example respondent described, a printed circuit board via placed near a component pad enabled leakage current in certain environmental conditions. The leakage current caused signal offset drifting, and false reading of the measurement was shown to the end user.

Another example case respondent described was related to digital master query slave response based digital bus of multiple detachable units. Connection of a unit to interface bus sometimes caused interruption of the communication on the whole bus and reset and restart of each connected unit.

Third case example respondent described was related to embedded system processor serial communication and was similar to the second case. In a modular system, the module interface was based on bus master querying data from slave devices. The bus had strict timing requirements, and the slave had to be able to reply fast to queries coming from the master with regular intervals. First devices built for the system interface had an embedded microcontroller which had special HW and low-level SW features for handling the interface listening and responses. This legacy interface bus timing and synchronization has shown to be problematic to handle with new processors. Handling the bus traffic listening and other tasks the module is supposed to do simultaneously requires hard real-time software. Any anomalies in regular communication may cause a problem with hard real-time software.

Could the defect have been discovered in automated system testing?

The failure from the first case example was detected in a production burn-in testing. Automated system testing could detect similar defects, but it might require equipment to control environmental conditions. It is better to try catching defects due to leakage currents in type-testing concentrating on specific functional block or unit.

Respondent thought that the second case example could be easily testable in automated system testing. Test system could control additional simple equipment connecting and disconnecting the power supply and communication interface of the modules and observe the system recovers or adds the connected module to the system correctly.

What kind of test equipment would have been required to automate the test?

In order to detect failures caused by leakage currents in analog electronics, a type testing of multiple samples over extended temperature and humidity range would be needed. Also, design engineer experience and checklists for design reviews are needed to avoid similar problems as in the case example.

Respondent had the following ideas for test equipment related to the second case.

- Test system should have a device which is capable of causing interruptions on communication and power supply.
- A device capable of causing fluctuations to supply voltage level could give useful information on the system robustness.

Based on your experience, what kind of measurement techniques should be considered to improve automated testing?

Respondent gave several ideas where automated testing could help improving testing coverage and revealing underlying problems. Ideas to consider were:

- More thorough type testing over temperature and humidity to reveal problems on analog circuits.
- Removable modules interface buses connection and disconnection testing to reveal timing problems.

- Testing of removable module recovery after a failure has occurred in communication signals (data and jitter).
- Mains power disconnection and battery removal/reconnect testing.
- User interfaces audio and visual items functionality, order and delay automated measurement.
- Device inspecting items from display automatically.

Note from the interviewer related to the last bullet:

Visual information and data on display could be studied during an automated test with many techniques. These are for example:

- The display data could be captured with test equipment and analyzed offline with screen-scraping techniques.
- Specific items from the screen and user interface could be monitored real-time with RGB sensors.
- Cameras and machine vision system could be used to detect items, text, and numbers from the screen.

Respondent 3

What kind of hardware related defects have you faced during your career?

Respondent gave several examples of issues which were revealed in system level or engineering testing.

First case respondent described was a product which has a feature to automatically re-start the device after power loss. The re-start function was not functional if the power-loss was short. Re-start from more prolonged power loss was functional. The root-cause was system power control unit which required power supply voltages to drop at low enough level before the power-on cycle was allowed.

The second case the respondent described was related to battery charging. Battery charging circuit had a maximum charging time safety limit. The charging circuit had a design error and the elapsed charging time was never reset after the charging cycle was considered complete in system software. In a device which was mainly AC-mains powered and kept the battery full using short charging cycles, multiple short charging cycles cumulated to the timer and which eventually reached the maximum limit. If the charging circuit reached the maximum charging time limit, the circuit assumed failure and disconnected the battery cells as a safety precaution.

Third case example causing trouble was related to power supplies of modular systems add-on modules. Modules were sometimes designed without keeping in mind the in-rush current during hot-swap connection. The inrush current of one module was so high that overcurrent protection in the host device was triggered and power from the whole modular system was cut-off. In the module connection also the data interface buffers have had design errors and a hot swapped module has caused data communi-cation interruptions before it has been powered-on properly and buffers set to correct state.

A fourth case example was related to a system which had multiple inputs for starting the device or initiating a power-off sequence. In some cases, software reading the input was not designed to perform the correct actions when the system was started from one

input and shut-down from another. This caused unexpected behavior during system shut-down.

Could the defects have been discovered in automated system testing?

The first case could have been quickly revealed in automated system testing if the test system had a mechanism to control the input power and vary the ON- and OFF times.

The second case could have been discovered as well in automated system testing. However, the respondent explained it was a design error in individual integrated circuit and should have been discovered in type tests of the function.

Automation of the third case should be easy. Test system should have the capability to control each device input power supply in the modular system individually. Testing different combinations or patterns of connections and disconnections of power supplies or data interfaces should then be possible.

Also, the fourth case example testing could be easily automated. Automated test system should have access to any input capable of starting or shutting down the system under test.

What kind of test equipment would have been required to automate the test?

Automated test system should have devices to control the connection of the input power supplies. Devices with a battery back-up should be monitored to see the device functions correctly from the battery or any of the multiple batteries. This would require control of each input power supply source separately.

Any user pluggable device connection and disconnection testing could be implemented with a test system controlled additional break-out devices.

Based on your experience, what kind of measurement techniques should be considered to improve automated testing?

Power ON/OFF/ON sequencing and variation of the OFF-time or ON-time with test system-controlled devices could be used to find problem cases. The variation in the OFF-time and ON-time can also reveal problems of hardware system controls for power supply start-up and shutdown sequencing.

Automated test system should also have access to any input capable of starting or shutting down the system under test.

Remark from the interviewer:

The start-up and shut-down sequences measurement automation could be done as a type test and repeated over device operating and extreme temperature ranges. In this case, measuring of each power supply and control signals would be required. Moving this kind of measurement to automated system testing could be possible and could be used to check the impact of the software changes in the power management system.

Respondent 4

What kind of hardware related defects have you faced during your career?

Respondent described a defect in a system with an internal mass memory module connected with a USB interface to a processor. The same internal mass memory is used in several product models. It was reported that in one product model the memory data gets corrupted. The failure occurs randomly and is challenging to reproduce. The failure could be due to the mass memory sharing a power supply with the USB ports external to the device. Failure has been tried to be reproduced by generating random loading to the external USB ports and observing the mass memory functionality. Also, the device has been put through temperature cycling while observing the data reading and writing.

Another example the respondent gave was related to interference between two analog measurement circuits. The original design of the product was made with a PCB with rigid-flex construction. The construction was expensive, and during the product life cycle, a design change was made to reduce the cost. The rigid-flex PCB was replaced with the flexible board, rigid printed circuit board and a board-to-board connector in between them. After the change, it was noticed that in some specific combination of use scenarios of the two analog circuits the other got interfered by the other circuit.

Could the defect have been discovered in automated system testing?

The respondent opinion was that the failure with the internal USB mass memory could not have been revealed in the system level automatic test as it is difficult to reproduce. The failure with the mass memory corruption would require component level stress testing with several units to get confidence in the hardware and allow statistical analysis of the failure rate. So instead of system testing, a type test for the failing component and interface would be a better testing method in this case.

The other example about the two analog measurement signals interfering each other could have been revealed in automatic testing. The changes in existing products should be tested very carefully to make sure there are no new problems introduced due to the design change.

What kind of test equipment would have been required to automate the test?

In the debugging of the internal USB mass memory failure, oscilloscope and signal generator and switchable load were used. The switchable load was controlled by the signal generator, and an oscilloscope is used to monitor the supply voltage. The oscilloscope trigger is taken from the signal generator. With this arrangement, the exact time of the load switching is captured in the supply voltage measurement. Together with the SW read/write a test to the USB mass memory, and the HW measurement the causality between memory corruption and supply voltage noise is tried to be captured.

For regression testing, automated testing which can go through various system states and functionalities combinations would be useful. Regression testing is important for verifying the HW changes made later does not introduce new problems. In the analog measurement case the respondent explained, the automated testing could have been such, that both signals are operated simultaneously and their ranges during the test are changed. This way the different combinations of the signals and their interaction with each other could have been tested automatically. Furthermore, if possible, the automated test system could examine the raw, unfiltered data from the analog sensor instead of the averaged and smoothed data shown or sent to the user.

Based on your experience, what kind of measurement techniques should be considered to improve automated testing?

The respondent opinion was that any test automation could improve the testing coverage and confidence on the HW. The possibility to run tests automatically overnight and weekend makes it much easier to run also test cases which would not be run otherwise.

Test automation for HW can be effortless if it is implemented at the functional block or unit level. With the same system, one also has the possibility to run automated tests over different voltage and temperature conditions.

Other essential lessons the respondent brought up was the power cycling and start-up tests. Automating such test makes it easy to observe the system recovers and works as intended after sudden power loss.

Respondent 5

What kind of hardware related defects have you faced during your career?

Respondent gave examples of the problems found on the development of a mobile data connectivity device.

The first problem the device had was a problem with power supplies. A problem occurred in low receiving signal conditions and with low battery voltage when the modem was turned on. In this corner case of low battery voltage and low receiving signal conditions of the network caused the modem to send registration to the network with highest power level, which caused the supply voltage levels to drop and reboot of the device. The problem was found in field testing and when the field test crew was returning from the network connection and roaming test route driven with a car. There was a certain point in the route where the receiving signal level on the network was low, and the battery level was low only during the end of the day after the device had been operating for a longer time.

Another example the respondent gave was related to the same mobile data connectivity device and the battery charger IC used in the device. The battery charger IC had a bug which appeared when the charger IC had been connected to DC-supply for three days. In this case, the charger IC stopped charging the battery and which then slowly drained empty. Workaround to fix this bug was to reset the charger IC every 5 hours to keep the IC functional while the device was in standby and connected to the power supply. The problem was not discovered until the system software was mature enough and long-duration tests were possible to run.

The third example from the same mobile data connectivity device was a problem of battery fuel gauge drifting. This problem appeared after the introduction of a new PCB layout and wasn't discovered until the product was in use by the customer. There were no specific regression tests defined which would have been run for each design change.

Could the defect have been discovered in automated system testing?

Respondent thought that the mobile data connectivity device problem with power integrity could have been discovered in automated testing easily. Automated test equipment could have run automated connectivity tests with different radio signal levels and with simulated battery levels.

The charger IC problem was discovered in system level testing, and if some long duration standby testing had been in place, the problem could have been discovered in an automated test as well.

The fuel gauge problem the respondent described could have also been discovered by an automated test if such had been in place.

What kind of test equipment would have been required to automate the test?

The mobile data connectivity device development was made in a start-up company which did not have much resources and equipment for automated testing. Also, the modem vendor had changed during product development. Respondent explained the equipment needed to automate the first example case would have been a radio network emulator and a battery emulator. With these two devices and simple test cases over battery voltage and signal level could have been created and run automatically with different product revisions, software and with multiple units.

The second case with the battery charger IC would have required only a voltage measurement from the battery during an extended duration test while the device is plugged to the charger. The battery voltage dropping even if the charger is connected would have been observed easily.

Based on your experience, what kind of measurement techniques should be considered to improve automated testing?

Respondent thought that regression testing of different HW-units could benefit a lot from test automation. The examples the respondent gave were problems which could have been discovered already at the unit or block level testing of the HW.

Based on the respondent experience and lessons learned from previous projects, the current project he is working with pays much attention to the device start-up and connection and disconnection test cases. These tests are run with different input power supply parameters and repeated even a thousand time to discover any underlying problems.