Developing Software for Mobile Devices: How to Do That Best

Invited Panel

Hermann Kaindl[1], Roberto Meli[2], Andreas Kurtz[3], Bernhard Bauer[4], Petre Dini[5]
[1)] TU Wien, Institute of Computer Technology, Vienna, Austria
[2)]DPO Srl, Italy
[3)]BMW AG, Integration Electric/Electronics, Software, Munich, Germany
[4)]University of Augsburg, Institute for Computer Science, Augsburg, Germany
[5)] Concordia University, Canada | China Space Agency Center, China

Emails: {hermann.kaindl@tuwien.ac.at, roberto.meli@dpo.it, Andreas.Kurtz@bmw.de, bauer@informatik.uni-augsburg.de, petre@iaria.org}

***Abstract*—Including computers and applications into mobile devices creates a major break-through in the applicability of computing systems and in the impact this had on users and even the society. While software development has always been costly and challenging, it is even more challenging for mobile devices. This raises the important question of how to best develop software for mobile devices.**

Keywords-mobile device; software development; user interface; Apps; testing.

I. INTRODUCTION

Mobile devices are comparably new and have differences to more traditional computers like mainframes and PCs (Personal Computers), such as the following:

- Different and possibly adaptive mobile user interfaces
- Context-aware/context-sensitive mobile applications
- Ubiquitous interactions, e.g., with wearables

Because of these differences, especially the software development for mobile devices poses challenges beyond that of traditional software development. This raises the important question of how to do that best.

The remainder of this paper is organized in the following manner. First, automated tailoring of user interfaces for smartphones and tablet computers is sketched and discussed in the context of mobile devices. Then, Apps development for mobile devices is contrasted with software measurement. After that, test automation is presented for cars viewed as mobile devices. Finally, challenges on designing and testing both Apps and wearable devices are presented.

II. TAILORED USER INTERFACES FOR SMARTPHONES AND TABLET COMPUTERS (HERMANN KAINDL)

A fairly obvious difference between, e.g., PCs and mobile devices such as smartphones and tablet computers is given through their relative screen sizes. Simply looking up a Web page prepared for a screen of a typical PC from a smartphone reveals problems like a tunnel view, which impair the usability. Sites looked up very often like those of CNN or airlines; therefore, they present their content *tailored* for large or small screens, respectively. This means extra effort for preparing these Web pages twice. In fact, there is a whole spectrum of screens sizes due to the large variability of screens of tablet computers and smartphones. When tailoring for a larger number of screen sizes, even more effort is required.

This issue calls for support through automation. In fact, technology exists for automated generation of Graphical User Interfaces (GUIs) [7][12][13]. In particular, also automated tailoring through optimization techniques is available [15]. Sample GUIs created (semi-)automatically can be viewed online:

- A demo flight booking GUI, see [1]
- An accommodation booking GUI, see [2], reverse-engineered from a real-world site (which is not online any more)

Of course, GUIs cannot be generated through magic. This approach requires high-level Discourse-based Communication Models [7][13] as well as (simple) device specifications to be created manually. While the effort for creating such models may not always pay back for generating GUIs of a single device, it most likely will for generating GUIs for multiple devices from a single model.

Unfortunately, the usability of fully-automatically generated GUIs is insufficient at the current state of the art.

 ISBN: 978-1-61208-458-9

So, we devised the so-called Custom Rules for addressing usability problems in a persistent way, which even showed that such rules can, in principle, be reused for multiple devices [16].

Still, there are obstacles for a wide-spread applicability of such an approach. Recently, we removed the problem of persistently including Custom Widgets through a GUI designer. The flight booking application [1] includes a seat picker widget as usual in real-world applications but unavailable in usual widget libraries.

This approach for automated tailoring even allows choosing different strategies such as tabbing or vertical scrolling, when the content does not fit the given screen size [15]. We found some evidence that the more wide-spread vertical scrolling is more efficient for use [14].

With respect to different screen sizes, we found some evidence that a user is typically more efficient on screens of larger sizes [17]. Of course, there is a trade-off with the mobility of such devices.

III. APP DEVELOPMENT & MEASUREMENT: ALLIES OR ENEMIES? (ROBERTO MELI)

Mobile application engineering is a relatively new branch of software engineering. Mobile application development and maintenance are characterized by:

- Small project sizes and short schedules
- Volatile scope
- Use of diverse technologies,
- User interface and user experience relevance
- Multimedia integration
- Geographical information integration
- Social remote and local interaction

These elements require an organizational approach based on:

- Time responsiveness
- Agile or evolutionary processes
- Small and very integrated teams
- Strong user involvement
- Interdisciplinary skills
- Supportive architectures and tools

Due to the deadline and uncertainty resolution focus and production orientation, teams are usually not too interested in "traditional" engineering practices, especially in measurement activities. They are perceived as "overhead". If any measurement is taken in the App project it is often a technological measurement.

A. Useful or not?

Nevertheless, "Functional" and "Non-Functional" Size Measurement Methods might be very useful in circumstances like the following:

- Corporate context
- Tender / Contract Management
- Project oriented development
- Prioritized and variable resource allocation
- Internal User driven
- Project productivity assessment needs
- Cost control emphasis

On the other side, measurement is not particularly significant in these situations:

- Personal context
- Informal internal contracts
- Service oriented development
- Self-managed team management
- Fixed resource allocation
- Market User driven
- Business Unit productivity assessment needs
- Time to market emphasis

When we consider Apps development effort, duration and staff estimation, apparently, there is no spread adoption of formal methods. Expert judgment seems to be the most adopted strategy. Unfortunately, the quality of these estimates is dependent on the quality of the estimators and many times it is impossible to compare different situations and to share expertise among different teams [1].

B. Typycal Processes & Deliverables

The process to develop an App is not so different from those applied to multimedia product or web-based applications [19]. A typical process should be:

- Agile-oriented
- Iteration-oriented
- Supported by tools

and should include phases like the following:

- Feasibility Study
- Collection of Functional and Non-Functional requirements
- App Wireframe creation
- Target architecture definition (Android, IOS, etc.)
- Back end
  - Defining the back end structure
  - Management of users
  - Server side logic
  - Customization of User Experience
  - Data integration (remote/local)
  - Push notification services
- Front end
  - Caching of data
  - Synchronization of App data
  - Mock ups Wire framing
  - UI design and development
  - UI improvements
  - Testing
  - Deployment

Developing an App is "project-oriented" but maintaining it may be "service oriented" with a continuous improvement process in place.

"The biggest issue, in mobile application development, still seems to be the diversity of platforms and devices.

 ISBN: 978-1-61208-458-9

Offering an App, be it enterprise specific or publicly available, means to provide different versions at least for the most widespread platforms (e.g., Android, Apple iOS, BlackBerry OS), operating system versions (with each version providing new functions or even altered appearance) and device types (with different display sizes and resolutions, controls and navigation styles). Since no standard cross-platform development approach has emerged so far, this plethora of combinations results in considerable development effort." [1]

Deliverables are documents and products in the multimedia domain and developing an App is not only a matter of Programmers and ICT people.

C. Which Measurement and Models?

Any adopted measurement model should be:

- Light
- Quick
- Simple
- Used by developers
- Complete
- Standard
- Product-oriented
- Easy to learn

Simple Function Point [21] has these characteristics for functional sizing. Measurement should be used for the governance of the process and the relationship among the different stakeholders.

In order to estimate effort, duration and staff, a complete model should be used which takes in account not only the functional requirements, but also the non-functional and process requirements like the one presented in [10].

IV. Software Based Test Automation Approach Using Integrated Signal Simulation (Andreas Kurtz and Bernhard Bauer)

New operating concepts are pushing from the Consumer Electronics Sector (CES) in the automotive industry. This change characterizes the development in the automotive industry and makes vehicle manufacturers increasingly become software developers. Software is an enabler for flexible and fast growing innovations. Especially the development cycle in the CES challenges the automotive industry not to lose connection. Vehicles nowadays must be linked with the customers' mobile devices and so become a mobile device. In today's vehicles, classic switches have almost become obsolete. "The automobile is the ultimate mobile device." [22]. Modern vehicles can be considered as mobile devices with Human-Machine Interfaces (HMI) such as displays, touch screens, gesture control and sensor operation. With increasing networking and alternative control options of functions, this change confronts the testing of customer functions of a vehicle with enormous challenges.

A. Challenges

Developing suitable software testing methods is the main challenge in software development for mobile devices to get high quality software. The speed of the hardware development, and software development cycles of the consumer electronic industry infuses the automotive industry. Because of changing trends, the growing networking of systems needs an innovative approach to be able to test the developed software fully automated. Innovative automation methods are a key part to handle the time pressure. In order to meet this challenge needs, a software-based approach with possibility to test the entire chain of reaction. A software-based approach allows reacting flexibly and fast on changes of software, especially changes on the interfaces.

Particularly, in the field of HMI, the technology is changing increasingly towards sensors without mechanical haptics. From the perspective of the user, the sensors and actuators on the HMI are fused to one single interface, touchscreens or sensor areas. This helps the designers to reduce costs because of being able to change the visual surface via software.

Further steps for interacting with the mobile device will be contactless input, gestures or the so-called air touch technology [9]. Following the term 'mobile devices' includes vehicles or subsystems of a vehicle.

The changing types of sensors with the innovation speed lead to new automation methods, to a software-based integrated approach being able to be adapted as fast as the software and hardware changes. Software-based integrated testing methods are missing due to consistent approaches, and lack of standardization. Especially in the automotive industry, software does not have a common architecture. This causes special/customizable solutions for each implementation.

B. Status

As mentioned before, an automobile becomes a mobile device. Depending on the point of view, the vehicle system is a mobile device second order. This means it is a distributed system combining severally mobile devices to a bigger mobile device. This consideration is possible because of the comparable basic architectures of networked Systems-On-a-Chip (SOC) or on the automotive domain networked software components on Electronic Control Units (ECUs) being SOCs. To show current solutions for automated testing these are separated in external- and software-internal solutions. With focus to model based testing methods [18], the testing is separated in four testing steps, in hierarchical order, and refers at each solution.

- Component test
- Integration test
- System test
- Acceptance test

1) External Automation Solution

Test automation with external automation solution makes only sense at component test, integration test or part system test (part system is a system cut in domain systems e.g., power-train system). However, the effort to adapt the interfaces increases enormous at part system test. Automation solutions for testing customer functions are

stimulating the component with physical hardware signals or remaining bus simulation.

2) Software-internal Automation Solution

A different solution, usable at any test layer, are additional software functions for an interaction in the software to trigger customer functions. This allows switching values or triggering customer functions, but needs for each customer function a custom-developed and integrated additional function. If the customer function is changed or moved to another hardware the additional function for software-based interaction has to be changed, too. Duplication of effort, in conjunction with increasing probability of errors may result.

C. The Methodological Approach

Figure 1 shows the methodical approach. Projecting this approach, based on an AUTOSAR [6] architecture, to other software architectures is possible. Various intermediate steps create a system model and test model for the requirements. Integrating an additional Software Component (SWC), called SIMulation Agent (SIM Agent) and deploying it to all ECUs, generate a software-based distributed simulation, with the help of an extended driver module to get access to the new simulation module. This allows simulating signal sequences in the driver layer with the advantage of reduced data types and a standardised interface. All other steps of the methodology are automated. From the test model, abstract test cases with abstract interfaces are created. These abstract interfaces become specific with the help of the deployment files. This allows performing the same test cases on various hardware platforms by adjusting the mapping 'Config', e.g., testing the same software on different mobile phones.

D. Alternative Proposals

An alternative approach could be a different interaction layer for this kind of simulation approach to avoid changes in the AUTOSAR architecture used in automotive domain. This is more compliant to the actual AUTOSAR standard but increases the number of data types.

V. CHALLENGES ON DESIGNING AND ON TESTING FOR WEARABLE DEVICES AND APPS (PETRE DINI)

Three complementary activities are specifically identified in new market communications activities, namely building wearable devices, designing Apps dedicated to them, and testing the solutions. The challenges are driven by several specific features characterizing each of them, but also by the nature of services they are used for and the human behavior. As some of the services are related to life threatening, testing the systems becomes a cornerstone process. The diversity of the devices, the heterogeneity of platforms, the absence of specific APIs and the scattered nature of system parts add to the complexity for verification and validation activities.

There is a continuously growing market boosted by Apple Watch very recently. Analysts predict a 42% growth for the wearable market within the next 5 years, while the Apps market should follow [3].

A. Challenges in Apps Development

The challenges faced by Apps developers are essentially induced by the wearable devices.

1) Devices and Apps

Some of the devices have always the screen on (like Peeble) that should be considered when designing an App to save as much energy as possible. Multiple screen sizes and formats (round, squared, e-paper display) need a fully adapted User Interface (UI) design. Computation options should also be limited to the minimum needed, as developers face limited computed power on a wearable device.

Wearable software is fragmented is more visible than for handheld devices is its intended purpose. Because of lack of established API, all coding of features takes place individually. So far, no accepted development cross-platforms exist; there are several operating systems, but no industry standard. There are ongoing industrial activities: Google is developing their Android wearable software development kit, NTT Docomo's Device Connect WebAP, GitHub is sharing the API as open software to enhance both technical specifications and API for mass commercialization.

There is a tendency to simply re-implement everything in the existing App on the wearable from an existing mobile App. This is not a recommended approach, as the interaction with the wearable watch is different that the interaction with a phone device. As a result, appropriate methodologies and guidelines should be developed and adopted. The current development platforms have limited features for an appropriate animation.

Troubleshooting wearable devices and Apps together leads to time-intensive development process and this is due to the frequency of troubleshooting on the new platforms.

There is a market push for reaching harmonization for Apps development. Juniper Research estimates the health related wearable devices industry will reach $53 billion in four years [4]. As a result, there is a potential that standardization and methodologies see a quick development. The finance sector is also helping, e.g., the introduction of Apple Pay along with the Apple Watch are current solutions; even more, payment-capable bracelets are offered by CaixaBank and Barclays.

The growing segment in the Apps marketplace will need a support for security and privacy. Practically, an embedded approach of wearable devices and Apps is a vital solution.

2) Thermal Considerations

A specific aspect is that wearable devices introduce some unique thermal design challenges that should be considered for devices, Apps and the entire system. This is not only referring to operability, but also to a required comfort level for humans. This design challenge is mainly for processor-intensive applications and units with complex displays.

According to Heussner [8] "electronics placed in direct contact with the skin need to maintain an ideal operating temperature at or below the core body temperature of 37°C (98.6°F). Anything above this is generally considered to be uncomfortable and hot (see Figure 2). Transitioning to much higher heat (above 40°C or 104°F) will trigger discomfort and pain for the wearer."

 ISBN: 978-1-61208-458-9

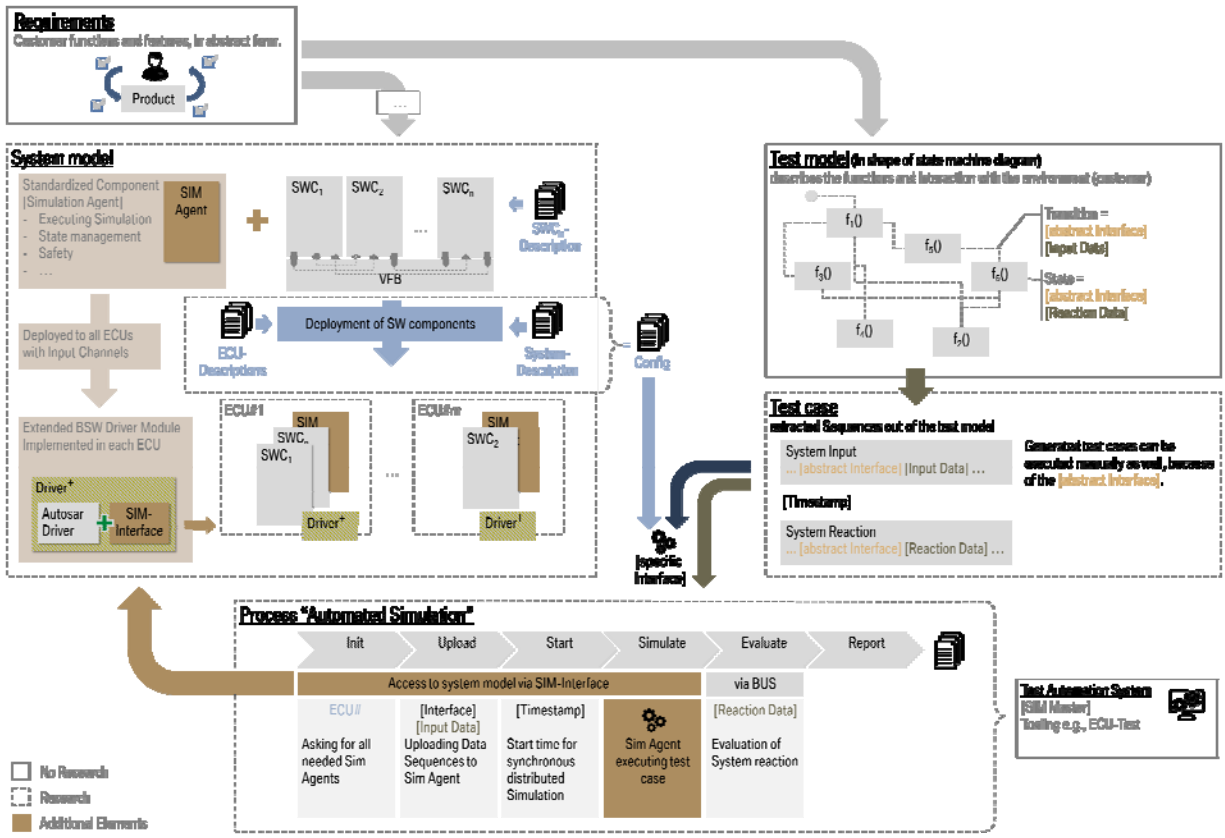

Figure 1. Software-based methodology for test automation in distributed systems.

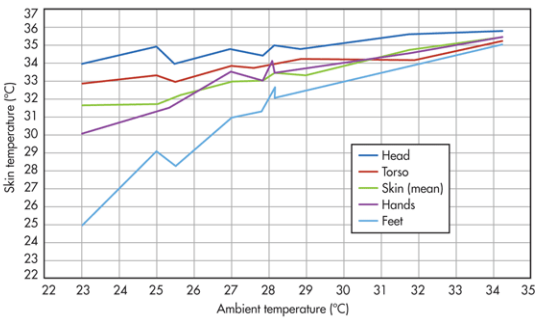

Figure 2. Thermal Considerations [8].

3) Materials and Environment

The design should address issues related to material interaction, reliability of interfaces, and impact on the thermal environment (for devices, Apps, systems). Chemical and a mechanical material interaction have to be calibrated; testing to optimize the package, the coating, and encapsulation is needed.

B. Testing Challenges

1) Testing Wearable

Wearable devices are deployed everywhere, with various functions, such as sensing, computing, transmitting, alerting, etc. A few characteristics make testing challenging, as listed below.

2) Small Screen

The designers must redefine the wearable screens and adapt their designing skills to miniaturization; dimensions should be carefully decided, as every pixel matters. There are certain limits at which a screen can be squeezed, yet being conveniently useful. Little of known UI/UX methodologies can be reused in designing new APIs.

3) Functional Testing

A big testing paradigm change was identified when the mobile devices arrived. Wearable devices comprise also different sensors and specific interactions that cannot be functionally tested by using traditional methods

4) Interaction

Testing should consider a myriad of sensor interactions. The large spectrum of interactions (Bluetooth, WI-Fi, hardware) leads to large coverage needs.

 ISBN: 978-1-61208-458-9

5) Battery Life

Energy and battery-based operation raises special maintenance issues and a real challenge for both wearable App developers and testers. These need also suitable testing criteria tuned to the new features of devices and Apps.

6) Testing for-Real

As wearable devices are quite specific, simply substituting them with emulators is not suitable; as the disciple is evolving in a rapid pace, trusting the results of such emulator is doubtful. Still, there are a few wearable on the market, e.g., Tizen, Android, etc.

7) Materials-oriented Testing

Due to metal migration concerns, biased testing is increasingly important to validate sensitivity in moist environments and to validate the risk of tin whiskers [8].

8) Testing body-wearable systems

There is a large variety of wearable devices and Apps, from fitness bands (which are essentially data collectors) to portable heads-up display; additionally, complex interactions occur between the touch display, cameras, and fast data communication with mobile platforms (see Figure 3).

Figure 3. Body wearable networks.

Complementary and specific components, like smart e-textiles, integrate stretch, pressure, and contact-based sensor elements, integrated within the fabric itself. Testing these components and their interactions requires appropriate experiments and calibration; this includes thermal aspects and materials characteristics on top of standard development guidelines of mobile devices and/or classic software development process.

VI. CONCLUSION

Of course, we cannot ultimately clarify how to best develop software for mobile devices. Still, we present a few related viewpoints that should help to pave the way towards a better understanding.

For instance, it is clear that different mobile devices need different user interfaces. With regard to screen size, automated GUI generation with automated tailoring may become an option.

Even whole cars may be viewed from the perspective of mobile devices today, since the automotive industry is increasingly influenced by the consumer electronics industry. This requires software-based integrated testing methods in order to keep up with the development.

What is specific on designing and testing wearable devices and Apps is that user experience is more relevant than in traditional approaches. It is a challenge to develop and test very specific features; e.g., “smart watches have very small screens and almost no buttons, making the use of space, navigation and user interaction incredibly important” [5].

Overall, it seems as though there will not be any single approach for developing software for mobile devices “best”.

REFERENCES

[1] http://ucp.ict.tuwien.ac.at/UI/FlightBooking

[2] http://ucp.ict.tuwien.ac.at/UI/accomodationBooking

[3] https://www.utest.com/articles/challanges-of-testing-wearable-devices

[4] http://www.foxnews.com/tech/2015/02/23/top-wearables-for-medical-issues.html

[5] http://www.belatrixsf.com/index.php/whitepaper-the-next-frontier-of-technology-wearables

[6] AUTOSAR Partnership, *AUTOSAR Layered Software Architecture*, 2014. [Online]. Available: http://www.autosar.org/.

[7] Falb, J., Kaindl, H., Horacek, H., Bogdan, C., Popp, R., and Arnautovic, E., A discourse model for interaction design based on theories of human communication. In *CHI ’06 Extended Abstracts on Human Factors in Computing Systems*, New York, NY, USA, 2006. ACM Press, pp. 754–759.

[8] Heussner, D. Texas Instruments, USA, http://electronicdesign.com/digital-ics/wearable-technologies-present-packaging-challenges

[9] Horn N., *BMW Group at the CES 2016 in Las Vegas.*. BMW presents the principle of the contactless touchscreen with AirTouch.

[10] Meli, R. A New Unified Model of Custom Software Costs Determination in Contracts, *Softeng2015*, Barcellona, 2015.

[11] André Nitze, Andreas Schmietendorf, Reiner Dumke, An Analogy-Based Effort Estimation Approach for Mobile Application Development Projects, IWSM-MENSURA, 2014, pp. 99-103, doi:10.1109/IWSM.Mensura.2014.9

[12] Paterno, F., Santoro, C., and Spano, L. D. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.* 16 (November 2009), 19:1–19:30.

[13] Popp, R., Raneburger, D., and Kaindl, H., Tool support for automated multi-device GUI generation from discourse-based communication models, in *Proceedings of the 5th ACM SIGCHI Symposium on Engineering Interactive computing systems* (*EICS’13)*. New York, NY, USA: ACM, 2013. Tool demo paper.

[14] Raneburger, D., Alonso-Rios, D., Popp, R., Kaindl, H., and Falb, J., A User Study with GUIs Tailored for Smartphones, in *Proceedings of the 14th IFIP TC 13 International Conference on Human-Computer Interaction - INTERACT 2013*, Part II, Springer LNCS 8118, Springer LNCS 8118, 2013, pp. 505–512.

[15] Raneburger, D., Kaindl, H., and Popp, R. Strategies for automated GUI tailoring for multiple device. In *Proceedings of the 48th Annual Hawaii International Conference on*

 ISBN: 978-1-61208-458-9

System Sciences (HICSS-48), IEEE Computer Society Press (Piscataway, NJ, USA, 2015), 507–516.

[16] Raneburger, D., Kaindl, H., and Popp, R. Model transformation rules for customization of multi-device graphical user interfaces. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '15, ACM (New York, NY, USA, 2015), 100–109.

[17] Raneburger, D., Popp, R., Alonso-Rios, D., Kaindl, H., and Falb, J., A User Study with GUIs Tailored for Smartphones and Tablet PCs, in *Proceedings of the 2013 IEEE International Conference on Systems, Man and Cybernetics (SMC'13)*, 2013, pp. 3727 - 3732.

[18] Roßner, T, C. Brandes, H. Götz and M. Winter. *Basiswissen modellbasierter Test*. dpunkt.verl., Heidelberg, 1 edition, 2010.

[19] Ruhe, M., Jeffery, R., Wieczorek, I., Cost estimation for Web applications, in *Proceedings of International Conference on Software Engineering (ICSE'03)*, 2003, 285–294.

[20] Seidl R., Baumgartner M., and Bucsics T. *Praxiswissen Testautomatisierung*. dpunkt, Heidelberg and Neckar, 1 edition, 2011.

[21] SiFPA, *Simple Function Point Functional Size Measurement Method*, Reference Manual SiFP-01.00-RM-EN-01.01, http://www.sifpa.org/en/index.htm, [retrieved: January, 2016].

[22] Diess, H., CES-Keynote, 2016

 ISBN: 978-1-61208-458-9