ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

Testing and Challenges in Modern Cyber Physical Systems.

Ricky Parmar*, and Mahek Merchant and Nishil Shah.

School of Computing Science and Engineering, VIT University, Vellore-632014, India.

ABSTRACT

Cyber Physical Systems (CPS) are depended upon and built from seamless unification of computational, communicational and physical components. In development of CPS we need secure and reliable software to bridge cyber world of computing and communication with physical world. It is challenging to integrate, test and verify the correctness of these components. For remote integration, testing, manufacturing and verifying correctness of CPS, the ability to capture individual levels of abstraction is critical. With the proliferation of smart embedded and mobile devices makes modelling and testing of modern CPS very complex which leads to many errors. So, we propose for agile iterative refinement processes on different levels of abstraction when any error occur in any CPS. We also show how modern cyber physical systems, which are highly depended on software, challenges traditional software engineering techniques and characteristics of modern cyber physical system.

Keywords: CPS, Agile Methodology, Cyber Physical System and Testing.

Corresponding author

ISSN: 0975-8585

INTRODUCTION

In order to get a better overview as well as the skill to fix more inconsistencies effectively, we need a suitable system model which will reduce overall effort and cost. However, several scenarios are taken into consideration, especially the designing of cyber-physical systems (CPSs) which describes physical components and its properties. Not even an exact model can replace for a real system. Numerous approaches on CPSs exclude a conceptual thinking level of the system description and lose the benefits of the conceptual description During the early stages of system development, platform independent architectural design has been recommended by researchers [11], [17]. The idea of pushing hardware- and software-dependent design as late as possible had been described by Sapienza et al. [11]. The focus is on comparison between adaptation and generalization of the software development methodologies. In our work, we augment these thoughts by joining combination of standard perspectives into the architectural levels with combination of system oriented perspectives into the design of system testing.

In the circumstances of standard-oriented aspects, we propose a testing strategy for CPSs which coordinates distinct conceptual levels of the system representation. The essential focuses for every distinct level are (1) whether we truly require the entire description of a system to break down its basic properties, and (2) identification of test cases required on this level. Much of the time, it is sufficient to describe some components of the system that are significant to a tangible reason. This methodology depends on refinement-based improvement of perplexing, intuitive systems [13], [14].

The above improvement can be considered as static: once the conceptual levels have been resolved, they stay settled all through the test arranging process. Practically, because of the complex and incorrect character of description, test designing, the analyser regularly commits errors, might return to the distinct levels of conceptual to make dynamic improvements. A fundamental hypothesis is hence required to see how such dynamic improvement of one level of concepts, influences remaining levels. This hypothesis can be then utilized as a premise for designing instruments supporting the system tester in such refinements.

On accounts for testing a CPS the "most inconsistent part" would be the tester. The principle of Agile Testing Design (ATD) is clear identification that the tester's action is not mistake-proof: mistakes can happen, both in the model and during testing, and ought to be taken into account. All the more solidly, locating of an error or inadequate data might bring about the tester to come back to the system and improve it, which in its turn might force further changes in the existing test arrangement. The word "Agile" is intended to mirror the iterative and incremental nature of the procedure of systems and test planning. We propose an Agile testing approach for cyber-physical-physical system, which consolidates two sorts of improvements: static (or system-oriented, intended to cover up unnecessary information) and dynamic (or tester-oriented, intended to provide the skill to remove error and finish the created antiquities). Designing tools for supporting this new prototype might build productivity of testing of CPSs and decrease test cost and time by agile prototype.

METHODS

Remote Testing of CPS's –

The important queries for a quality-oriented engineering in a worldwide context is which includes we have to check at which level of concepts. Testing, and additionally confirmation, at the tangible level is costlier than on a conceptual one, particularly if some particular alteration inside of the systems are fundamental. Sensibly, we should have more serious testing at logical level to decrease the general size of test suite for the following levels.

We have recommended to have three principle meta-levels of deliberation:

I. **Conceptual Level**, where we work on the analytical engineering of the system and a dynamic model of the environment, and test. The interoperability between consistent parts of our architecture.
II. **Virtual Level,** where identify software and hardware designs and work on both virtual and genuine representations of the hardware parts. On this level we test the interoperability in the middle of virtual and genuine systems.

III. **Cyber-Physical Level,** where we work on genuine systems parts, and test the interoperability between genuine systems that are physically present for testing.

One of the benefit of this methodology is congruity with the thoughts of Virtual Commissioning Technology [2], [7], which guarantees a more proficient treatment of the many-sided quality in systems. In our present work we apply on every conceptual level the ATD procedure, considering the error-prone nature of the tester's work, and supporting the tester in improving and boost test sets on every conceptual level. To expand the efficiency on the Virtual Level, we can utilize facilities as the Virtual Interoperability Testing Laboratory (VITELab, cf. [1], [19]), where the interoperability experiment and testing are performed early and remotely.

At some level we have to change from the immaculate conceptual (legitimate) representation of the system to a cyber-physical one, in any case, amid various improvement steps we test (and improve) the system or segment utilizing a virtual situation, and after that proceed with testing in a real domain.

When we check some system properties as excessively solid for the present specification layer and exclude them for keeping the model more discernible and theoretical, we need to watch that whether we free any critical data about the system, on this level of conceptual or as a rule. When we check a few system tests as superfluous/discretionary to expand effectiveness of the testing process, we need to check whether some essential system properties are not secured by the selected test set.

If the information is not relevant, it could impact on the general modelling result after a few improvement steps, i.e. at more tangible levels that are more close to the real systems in the physical world. Hence, indicating system we ought to settle on every one of the choices on deliberation in the model straightforward and track them expressly – for the situation of disagreement between the model and the real system this permits to discover the issue simpler and quicker.

Testing Structure for CPS-

Consider that generally any cyber physical system A is described completely by the set CHAR(A) of its cyber-physical properties. If we have n level of abstraction then at each level m of abstraction the characteristics of the cyber physical system CHAR(A) can be divided into two subsets, Let the characteristics reflected at this level of abstraction be denoted by set $MCHAR^m(A)$ and characteristics from which we abstract at this level be denoted by set $ABS^m(A)$, knowingly or unknowingly. The intentional abstraction of the characteristics of the system be denoted by $ABS_{know}{}^m(A)$ and are tracked during system development, $ABS_{know}{}^m(A) \subseteq ABS^m(A)$.

Following holds for any level m of abstraction:

1) $MCHAR^m(A) \cup ABS^m(A) = CHAR(A)$
2) $MCHAR^m(A) \cap ABS^m(A) = \varnothing$

Each test on corresponding level should cover each characteristics $c \in MCHAR^m(A)$. Then again, we don't have to determine on this level any tests to cover the characteristics from the set $ABS^m(A)$. Let set TST(A) be the set of test required on the level m, then it has to be generated from $MCHAR^m(A)$. Some characteristics are moved from set ABS to set MCHAR with each phase of fine-tuning. It can be said in some sense that the termination function of the modelling process is represented by set ABS: in case m corresponds to the real representation of the system, we get $MCHAR^m(A) = CHAR(A)$ and $ABS^m(A) = \varnothing$.

Number of assumption are used on environment of the system A on every level m. Consider set $EN_{AS}{}^m$ represents these assumptions. In exercise, we see the abstraction levels as relating to stages in an imperfect process rather than perspectives which are kept integral and reliable. If compared set $ABS_{know}{}^m(A)$ with $EN_{AS}{}^m$, it would be impractical to expect monotonicity between the cardinality of set $EN_{AS}{}^m$ and the number m. With each phase of fine-tuning some of the assumptions made about the environment of the cyber physical system would become unnecessary or weak, but also stronger version of some assumptions may be needed at next phase so that all of the characteristics of the system are met with. On each level of modelling it is imperative

ISSN: 0975-8585

to detect the changes in EN_{AS}, so that we are able to find out on which level, which characteristics of the system should be retested, if some contradiction between EN_{AS}^{m} and actual targeting environment is found on some fine-tuning step m+1. Hence, during the testing phase, collected assumptions should be verified and model should be altered according to the results of the testing if something is missing or improper.

Introducing Agile Testing Design into CPS Testing Framework –

Agile software development process requires agile testing practices and guidelines [3], [10], [22]. An ASDP (Agile Software Development Process) concentrates on providing early and quick production of working code. Iterative and incremental development of software is supported by corresponding ASDP models.

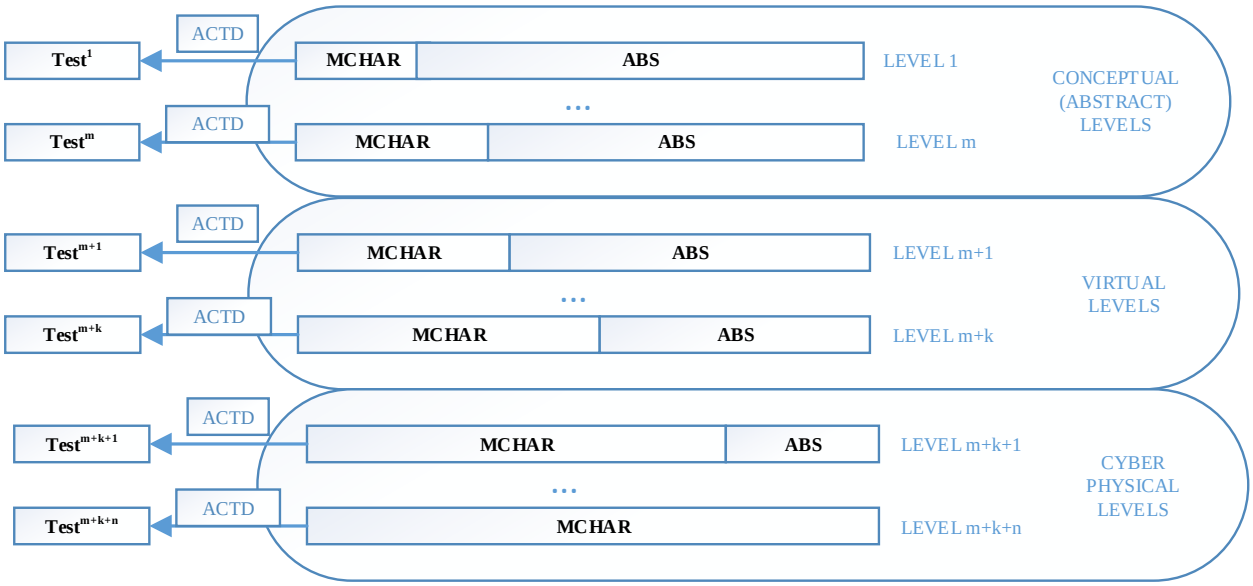

Figure 1- Different Levels of Abstraction for Agile Test Design

The idea of agile combinatorial test design was first introduced in context of combinatorial test design (CTD) in [23]. In CTD [24], wherein the space to be tested is called combinatorial model, is a productive test planning technique. Combinatorial model is characterised by a set of parameters, their individual values and restriction on the value combinations. At different phases and scopes of testing CTD methodology can be applied, including system-level testing, end-to-end testing, feature-level testing, service-level testing and application program level testing. With the goal to increase the architectural sustainability, in [18] it was suggested to use architectural-based combinatorial approach for testing of CPS. Architecture was given priority in CTD for cost-effective longevity and endurance. In this paper we bring agile combinatorial test design aspects for the development process.

Using a finite set of system parameters SP = { X_1,…,X_n } a system is modelled in combinatorial test design, let { $V(X_1)$,….., $V(X_n)$ } be their corresponding values. Our primary focus will be on interactions between the different values of parameters, where at most one value of each parameter might appear. Some value for each system parameters appears in an interaction of size n, consider this as our scenario. If for every $i \in I$ there is some $s \in S$, such that $i \subseteq s$, where S is set of scenarios and I is set of interactions, then we can say that a set of scenario S covers a set of interactions I. A combinatorial model $\mathcal{E}$ of a system comprises of sets of scenarios, where all test executable in system are defined. A test plan SP = ($\mathcal{E}$, I, S), where we already know that S cover I, I is also called as coverage requirement which is set of interactions, S is a set of scenarios which we can call as tests and $\mathcal{E}$ is combinatorial model. Pairwise testing is considered to be the most standard coverage requirement [8], because every (executable) pair of possible values of system parameters is considered here. If we write our test plan again in terms of pairwise testing then our test plan can be formulated as SP = ($\mathcal{E}$, I_{pair}($\mathcal{E}$), S), where set of all interaction of size 2 is defined by I_{pair}.

ISSN: 0975-8585

Generally, there are following phases in CTD methodology. First and foremost, combinatorial model is constructed by tester by providing executable set of scenarios in the system. Then we reach second phase after choosing coverage requirement which is to construct a test plan i.e. proposing a set of tests over the model, so that with regard to chosen coverage, full coverage is achieved. More precisely, tester's aim will be to provide a valid test plan SP = ($\mathcal{E}$, I, S). Plan is valid under following cases. First, all coverage requirement in I is satisfied by S. Second, S is subset of $\mathcal{E}$. While in case of standard combinatorial test design approach assumptions are made that first is satisfied before handling second, in agile combinatorial test design they are handled in parallel. Hence assumption of its availability is not made in our design. Rather, when a tester specifies a test case S it is extracted iteratively from it, as well as some logical constraints as defined above on the combinatorial model, which make available only partial information on $\mathcal{E}$. For this reason, in our system we divide the space of tests into three basic types so uncertainty is represented explicitly. (a) Authenticated (green): These tests are confirmed as executable by the tester (where tester uses some selected confirmation strategy.); (b) Excluded (red): These tests are rejected by tester because of them being irrelevant or impossible on current abstraction level (c) Uncertain (yellow): As enough information is not provided for classification of this tests, tester has not classified these test to be authenticated/Excluded. Minimization of this uncertainty will be our ultimate goal and this can be done by firing series of queries to tester so that we can authenticate/exclude tests. Example scenario: Consider a system with two robots in it namely RB1 and RB2 which are interacting with each other. Suppose our level of abstraction is 0, our system can be modelled as follows. Suppose that each robot has a gripper which has two modes (G1 and G2) they can either open or close to hold an object. Consider each robot can position (Q1 and Q2) its arm, which has a finite set of position value {post^1, post^2, post^3} used as abstract value. Suppose by means of pairwise coverage requirement, tester constructs a system model and a test plan. Consider a scenario where RB1 gives an object to RB2, this task can only be performed when the position of the gripper of both robots are same, also the gripper of RB1 has to be closed while that of RB2 has to be in open state. What if for our system, tester only provides logical state i.e. G1 = close and G2 = open, while omitting the information about the position by mistake or faultily. This will lead to system model in which all the tests are coloured yellow (Uncertain i.e. not yet confirmed by tester). The tester then goes on to construct set of tests {test1, test2}, removing faultily post^3. Where test1 = {Q1 = post^1, Q2 = post^2, G1 = close^1, G2 = open^2} and test2 = {Q1 = post^2, Q2 = post^2, G1 = close^1, G2 = open^2}. Now, these two tests are authenticated by tester and coloured green. As we can see pairwise coverage is not attained and tester's mistake is exposed at this point. Because, as an example one can see that interaction {Q1 = post^3, Q2 = post^3} remains uncovered. This type of mistake can occur if tester forgot to add some tests or due to the fact that tester considered non-executable tests as possible. Whenever a suitable query is popped-up, it may prompt tester to update logical conditions with Q1 = Q2 or extend the test plan by including {Q1 = post^3, Q2 = post^3, G1 = cloposse^1, G2 = open^2}.

Software Engineering Specifics for Modern CPS

Physical Distribution and connection of the large-scale with the physical domain makes modern CPS fairly particular as far as software engineering (SE). In this part, we outline these specifics from the point of view of SE presumptions and opportunities.

SE Impressions Violated in modern CPS-

Various impressions that are generally assumed in typical SE of general purpose software systems (GPSS) are overstepped in modern CPS. The suppositions made on the way that a great deal of multifaceted nature identified with the systems administration and environment can be in GPSS considered low-level and disconnected away by OS and middleware. It is but obvious, that even in customary SE some key assumptions might be damaged when creating GPSS with exceptional needs (e.g., high-availability, open-endedness). By and by, modern CPS are obviously particular in this setting by the expansive number of such disregarded presumptions.

In this manner, below we recognize and examine various suppositions in conventional SE of GPSS.

Static physical structure

Even if information and code are liable to portability in GPSS, the physical hubs where the code is

running are normally stationary. In modern CPS, the physical substratum is persistently advancing, as hubs move in the physical environment. The key test is the manner by which to outline continually changing substratum to the system of computational hubs so that stringent necessities on the sought administrations are dependably met.

Location unaffectedness

The expense and benefit of achieving a specific hub is regularly not altogether affected by its physical area. This autonomy encourages production of front-end and element circulated GPSS and is for the most part considered an advantage. In modern CPS, territory of peer hubs is a central configuration limitation, since physical nearness straightforwardly influences reachability and network on one hand and utilitarian accuracy on the other.

Reachability

GPSS regularly depend on the Internet system stack for the fundamental correspondence conventions. This implies with high likelihood any hub can effectively set up point-to-point correspondence joins with whatever other hub in the framework. In modern CPS there is no such ensure, as hubs frequently work over element systems without a perpetual framework. This confinement forces a crucial limitation in the outline of modern CPS, since hubs are required to work in full self-governance, even separated from their companions.

Stable links

In many GPSS, on top of having the capacity to reach and interface with remote subsystems, associations are commonly viewed as steady. This is showed in handling of correspondence mistakes in such frameworks: error is considered as special cases that must be taken care. In modernCPS errors in correspondence are the rule, not the special case. In this way, they can no more be taken care of as exemptions. The property of loose availability must be recognized and in a perfect world reflected in the utilized SE deliberations.

Accessibility of global state

Reasoning over the global position of an appropriated framework is a necessity for some applications. In spite of the fact that procedures exist for typically circulated GPSS, they are not straightforwardly physical to modern CPS. Additionally, since the nearby state in modern CPS develops ceaselessly with the physical environment, achieving global state for the most part is infeasible.

Marginality of ongoing features

GPSS commonly don't force hard ongoing requirements on their operation and correspondence. At the point when time, it is for the most part on the grounds that late reactions might block framework execution as opposed to rightness. In modern CPS, the progression of time turns into a focal element of framework conduct and plan, following stringent idea of time is key for measuring, anticipating and controlling properties of the physical environment.

Crisp consistency

In typical distributed GPSS, there is a fresh idea of information consistency: The information is either steady or not. Then again, in modern CPS, where strict dispersed synchronization turns out to be excessively costly, such elucidation of consistency is not alluring. Maybe, in modern CPS it is vital to evaluate and/or ensure a level of consistency.

Controlled dynamism

Many GPSS are vital as they powerfully adjust to changes and recoup from defame states. The perception is, however, that the dynamism is commonly an aftereffect of activities started by the framework

itself or its head. Despite what might be expected, in modern CPS dynamism is intrinsic, forced by the physical environment itself. Therefore, modern CPS need to distinguish and recuperate from unexpected and frequently unanticipated circumstances in their surroundings in a non-troublesome manner and without supervision.

Focus on behave

Outputs of a GPSS are ordinarily responses to express jolts. Rather than sitting tight for an occasion, modern CPS need to work ceaselessly. This implies the thought of "endeavoring to accomplish" is focal in framework conduct. In that sense, modern CPS need to always respond to furthermore perform changes taking into account properties that are either detected or anticipated. Depending on straightforward response designs in modern CPS is inadequate, since it might prompt motions and insecurity.

Stateful transmission

GPSS for the most part accept stateful correspondence in the correspondence conventions they utilize. This empowers powerful synchronization among conveyed parts. In addition, since stable associations are accepted, blunders are dealt with as extraordinary and recognized and understood by means of unequivocal mistake recuperation. In modern CPS, stateful correspondence does not scale.

SE Opportunities in modern CPS

As pointed out in 1.1, none of the talked about suppositions can be for the most part assumed in modern CPS. This makes it a non-insignificant test to create modern CPS by applying customary SE strategies. Be that as it may, it is inappropriate to see all specifics of modern CPS as hindering their advancement, since they might give chances to getting around the abused suppositions. In this point of view, it is attractive to exploit such modern CPS specifics as opposed to going for adjusting customary SE strategies. Once more, to pinpoint this thought, we have aggregated a rundown of specifics, which we accept can be beneficially misused in tending to the damaged suppositions. Regardless of not being finished, we trust this rundown still gives an essential exploration course for modern CPS plan strategies:

Physical versatility

Devices utilized as a part of modern CPS range from stationary to convenient and portable ones. Computational hubs conveyed on cell phones can convey data while moving. This adds to the general connectedness of the framework, as a portable hub covers a much greater physical territory while moving, and can viably spread the data in the region and associate generally disengaged system allotments.

Physical location

The way that gadgets in modern CPS are physically close gives a characteristic approach to segment the framework into subsystems in view of topographical area. This is, once more, unique to modern CPS; universally useful frameworks are seldom parceled in view of the physical area, as a result of the generally valuable presumption on area mindlessness. Having such a characteristic apportioning can be effortlessly abused to accomplish elevated amounts of adaptability.

Location-reliance of data

Data in modern CPS are regularly area indigent, implying that the estimation of certain quantifiable framework properties rely on upon the physical area of the sensors that give the crude information. This reliance, in blend with the physical nearness of sensor hubs, takes into account information sharing and reuse among close-by hubs and can possibly add to framework power.

Physical laws in data progression

Since modern CPS operation ordinarily includes detecting physical-environment properties, one can exploit the physical laws that represent the advancement of the estimations of such properties to

assess/anticipate their genuine qualities. As a result, a quality that is somewhat stale can in any case be utilized, if certain wellbeing limits on its development in time can be built up.

CONCLUSION

In this paper we presented our work on testing cyber physical system with agile test design methodology. We use here the idea of agile combinatorial test design and refinement based testing and take into account the mistake made by tester providing revision and refinement. The efficiency of testing cyber physical system can be ominously with our proposed tactics. While following agile paradigm, testing cost and time can also be reduced. In this paper we also made an effort to pin down the challenges and drawbacks associated by applying traditional software engineering practices to current software intensive cyber physical systems.

REFERENCES

[1] J. O. Blech, M. Spichkova, I. Peake, and H. Schmidt. Cyber-virtual systems: Simulation, validation & visualization. In 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2014), 2014.

[2] R. Drath, P. Weber, and N. Mauser. An evolutionary approach for the industrial introduction of virtual commissioning. In IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pages 5–8, 2008.

[3] O. Hazzan and Y. Dubinsky. The agile manifesto. In Agile Anywhere, pages 9–14. Springer International Publishing, 2014.

[4] T. D. Hellmann, A. Sharma, J. Ferreira, and F. Maurer. Agile testing: Past, present, and future–charting a systematic map of testing in agile software development. In Agile Conference (AGILE), 2012, pages 55–63. IEEE, 2012.

[5] H. Liu, M. Spichkova, H. Schmidt, T. Sellis, and M. Duckham. Spatiotemporal architecture-based framework for testing services in the cloud. In 24th Australasian Software Engineering Conference (ASWEC 2015), 2015.

[6] H. Liu, M. Spichkova, H. Schmidt, A. Ulrich, H. Sauer, and J. Wieghardt. Efficient testing based on logical architecture. In 24th Australasian Software Engineering Conference (ASWEC 2015), 2015.

[7] S. Makris, G. Michalos, and G. Chryssolouris. Virtual commissioning of an assembly cell with cooperating robots. Advances in Decision Sciences, 2012, 2012.

[8] C. Nie and H. Leung. A survey of combinatorial testing. ACM ComputSurv., 43(2):11:1–11:29, Feb. 2011.

[9] F. Redmill and J. Rajan. Human factors in safety-critical systems. Butterworth-Heinemann, 1997.

[10] B. Rumpe. Agile test-based modeling. In Proceedings of the 2006 International Conference on Software Engineering Research & Practice (SERP). CSREA Press, 2006.

[11] G. Sapienza, I. Crnkovic, and T. Seceleanu. Towards a methodology for hardware and software design separation in embedded systems. In Proc. of the ICSEA, pages 557–562. IARIA, 2012.

[12] I. Segall and R. Tzoref-Brill. Interactive refinement of combinatorial test plans. In Software Engineering (ICSE), 2012 34th International Conference on, pages 1371–1374, 2012.

[13] M. Spichkova. Refinement-based verification of interactive real-time systems. Electronic Notes in Theoretical Computer Science, 214:131–157, 2008.

[14] M. Spichkova. Architecture: Requirements + Decomposition + Refinement. Softwaretechnik-Trends, 31:4, 2011.

[15] M. Spichkova. Human Factors of Formal Methods. In In IADIS Interfaces and Human Computer Interaction 2012. IHCI 2012, 2012.

[16] M. Spichkova. Design of formal languages and interfaces: formal does not mean unreadable. In Emerging Research and Trends in Interactivity and the Human-Computer Interface. IGI Global, 2013.

[17] M. Spichkova and A. Campetelli. Towards system development methodologies: From software to cyber-physical domain. In First International Workshop on Formal Techniques for Safety-Critical Systems, 2012.

[18] M. Spichkova, H. Liu, and H. Schmidt. Towards quality-oriented architecture: Integration in a global context. In Proceedings of the 2015 European Conference on Software Architecture Workshops, page 64. ACM, 2015.

[19] M. Spichkova, H. Schmidt, and I. Peake. From abstract modelling to remote cyber-physical

integration/interoperability testing. In Improving Systems and Software Engineering Conference, 2013.

[20] M. Spichkova, X. Zhu, and D. Mou. Do we really need to write documentation for a system? In International Conference on ModelDriven Engineering and Software Development (MODELSWARD'13), 2013.

[21] D. Talby, A. Keren, O. Hazzan, and Y. Dubinsky. Agile software testing in a large-scale project. IEEE Software, 23(4):30–37, 2006.

[22] D. Turk, R. B. France, and B. Rumpe. Assumptions underlying agile software development processes. Journal of Database Management, 16:62–87, 2005.

[23] A. Zamansky and E. Farchi. Helping the tester get it right: Towards supporting agile combinatorial test design. In 2nd Human-Oriented Formal Methods workshop (HOFM 2015), 2015.

[24] J. Zhang, Z. Zhang, and F. Ma. Introduction to combinatorial testing. In Automatic Generation of Combinatorial Test Data, pages 1–16. Springer, 2014.