A Survey on Testing for Cyber Physical System

Sara Abbaspour Asadollah[1(✉)], Rafia Inam[2], and Hans Hansson[1]

[1] Mälardalen University, Västerås, Sweden
{sara.abbaspour,hans.hansson}@mdh.se
[2] Ericsson AB, Kista, Sweden
rafia.inam@ericsson.com

Abstract. Cyber Physical Systems (CPS) bridge the cyber-world of computing and communications with the physical world and require development of secure and reliable software. It asserts a big challenge not only on testing and verifying the correctness of all physical and cyber components of such big systems, but also on integration of these components. This paper develops a categorization of multiple levels of testing required to test CPS and makes a comparison of these levels with the levels of software testing based on the V-model. It presents a detailed state-of-the-art survey on the testing approaches performed on the CPS. Further, it provides challenges in CPS testing.

Keywords: Testing · Cyber physical systems · Survey

1 Introduction

Cyber Physical System (CPS) is one type of complex engineering systems, which is based on the integration of physical, computation and communication parts. The operation of this type of systems needs to be controlled, coordinated, monitored and integrated by a computing and communication core which is integrated in the physical environment. Examples of CPS with different functionality can be found in diverse areas, such as health care, smart transportations, data centers, smart buildings, smart homes, power grids and safety support system. The most important issues within these systems are dependability, efficiency, and security.

A CPS typically consists of *physical and cyber* spaces, with various sensors and actuators, as graphically shown in Fig. 1. These spaces integrate computation and physical processes, and are interconnected by a network layer for exchange of data between these two spaces.

Testing and verifying the correctness of all physical and cyber components of such complex CPS poses a big challenge. It may contain hardware and software testing, computation and communicational testing, extra-functional testing for each component individually, besides integration, and system testing to test the complete system. Despite the well-established concept of CPS, relatively little work has been performed on testing methods for CPS.

K. El-Fakih et al. (Eds.): ICTSS 2015, LNCS 9447, pp. 194–207, 2015.
DOI: 10.1007/978-3-319-25945-1_12

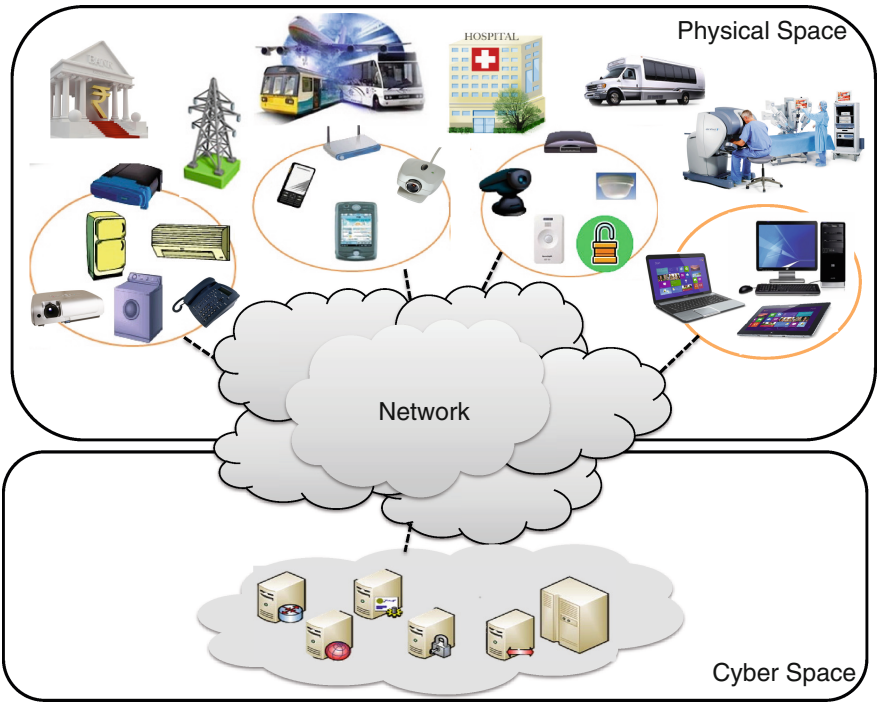

Fig. 1. Cyber physical system components

Contributions: This paper presents (1) multiple levels of testing required to test a complete CPS, and makes a comparison of these levels with the levels of testing based on the V-model; (2) a state-of-the-art survey on different testing methods performed on the CPS; and (3) challenges in CPS testing.

Outline: Section 2 presents related work. Section 3 presents different levels of CPS testing. Section 4 investigates state-of-the-art methods for testing CPS, Sect. 5 outlines some challenges in testing the CPS and Sect. 6 concludes the paper.

2 Related Work

Testing is an essential activity in engineering and it is widely used in industry in order to guarantee the quality of any type of system. Bertolino [4] has organized the outstanding research challenges for software testing into a consistent roadmap.

There are many studies performed on testing for different SW development methods [5,11,14,16] (like web services, cloud, software product lines), and testing software development process [24]. However, to the best of our knowledge,

this paper is the first study to explore the different CPS testing types and performing a state-of-the-art survey on it.

Koray, et al. performed a survey of software testing of cloud-based systems and classified related literature according to research activities performed in the cloud-based testing area [11]. They also identified and clarified the terminologies, the gaps and open issues.

Bozkurt et al., presented a survey on testing Web service by classifying the testing types and techniques [5]. They also categorized the previous work undertaken on web service testing based on some functional testing techniques. Another survey on web application testing is performed by Li, et al. [16]. They reviewed the recent Web testing advances and discussed the employed techniques, targets and goals. Further, they categorized Web application testing techniques into a number of groups (e.g., scanning and crawling techniques, search-based techniques, mutation testing and more).

Tevanlinna, et al., presented a survey on product family testing and described methodology and processes for this testing [24]. They emphasized the use of a careful planned testing process that can be easily adapted and used for product families in different application domains. They evaluated the current state-of-the-art in product family testing and highlight problems that need to be addressed in the future.

A survey framework on software product line testing (SPL) is defined in [14]. Lee, et al., divided the SPL testing into two separate test engineering activities: (1) domain testing and (2) application testing. In domain testing assets (test plans, test cases, and test scenarios) are used as inputs to application testing. They believe that in the normal case, complete products are not obtained during domain engineering sine domain engineering focuses on core asset development. Therefore, in most cases domain system testing can only be conducted in a limited way. By testing core assets in domain testing the application testing can focus on the application specific parts, which were not covered in domain testing. They explained that SPL testing has a W-shape lifecycle, which is typically called, extended V-model and formed by two overlapping V-models. They explored the software product line testing approaches by defining a reference SPL testing processes and identifying their key research perspectives which were related to testing field.

Research on using monitors for testing purpose has mostly focused on monitors for software that is neither real-time nor distributed. Only a few studies have addressed monitoring real-time or distributed systems, which characterized safety-critical systems such as flight-critical systems for aircraft and spacecraft. A survey on monitors to test distributed real-time systems [7].

Abbaspour et al. classified concurrency bugs based on the observable properties for multicore applications [1]. They address that such a taxonomy can help testers and debuggers to understand the causes of concurrency bugs and to avoid introducing them. Classification helps them to make appropriate decisions when they encounter problems. It can serve as a structure in which the current body of knowledge can be arranged, thereby allowing for identification of gaps in this

knowledge by easing the debugging process, and help users heuristics for more precise detection tools.

Gupta, et al. compare different approaches towards design and verification of energy sustainable computing for CPS [8]. They address that a perfect way to perform verification is either through experimentation on actual deployment of a CPS or through accurate simulation of the system. Simulation based verification is widely used since the resources required to build experimental test-bed may not be affordable. Both simulation and experimentation can also be used to characterize various functions. Moreover, in many CPS, verification is required at the design time without real deployment. The early design time verification has two advantages: it avoids creating real test-scenarios putting lives at risk; and it provides a way to guarantee and certify the CPS behavior.

3 CPS Testing

In this section we present different levels of testing required to check and verify CPS, and then compares these levels with the testing types of the V-model.

3.1 Levels of CPS Testing

As explained in Sect. 1 a variety of testing types may be used in testing the different components of a CPS.

Hardware Testing: Hardware testing consists of testing hardware components of CPS, including tests of each component's functionality based on the system requirements. The most common and important variables in evaluating and testing hardware such as desktops, laptops, printers, PDAs and other important hardware that are used in the CPS are memory size, speed, storage capacity, spindle size, I/O interfaces (ports), synchronization capabilities, expandability and similar.

There is another issue in checking the hardware functionality, which is hardware verification. Hardware verification tests the hardware under specific conditions, checking that the hardware follows the local environment requirements conditions, confirming to the applicable quality assurance measures and more. Thus, hardware testing is typically more structured and detailed than hardware verification. There are some similarities between software and hardware testing. Hardware testing has fewer steps and does not usually undergo a pilot project.

Structural and Computation Testing: Structural testing is normally based on the detailed design and not on the required functions of the program. However, for computation testing the tester (or developer) uses the structure of the program and chose paths that are used to recognize domains. Computation testing is one form of structural testing, which focuses on the computational part of software.

Extra-Functional Properties Testing (EFP Testing): Multiple parts of a CPS are often embedded systems executing real-time software. For such a

system it is required to guarantee both (1) function correctness and (2) non-functional or extra-functional correctness. Extra-functional properties are closely related to the inherent interaction with the system environment. Examples of such properties may be temperature, power consumption, and timing [21,26].

Network Testing: Communication and network testing are other important issues in a CPS testing. The aim of this testing is to check and verify the protocols used in a communication flow among multiple devices and users. Besides, the actual measurement and recording of a networks, state of operation during a period of time is called network testing. In network testing the tester is assisted for verifying, controlling or comparing the performance by recording the current state of network operation.

Integration Testing: Combining the individual software modules in a group and testing the grouped module is called integration testing. It is a phase of software testing and prepares the system for the next phase of testing which is system testing. The errors, that appear because of combining different units are detected in this phase. In other words, any inconsistency between integrated software units or integrated software units and hardware can be discovered by integration testing [19].

System Testing: Finally, hardware system testing or software system testing is related to test a complete integrated system when all units in a system are integrated to fulfil the overall requirements of the system [19]. The purpose of this testing is not just testing the design of the system, but to also test and verify the behavior and the assumed expectations of the user.

3.2 Comparison of Testing Levels of CPS with the Traditional V-Model

Comparing to the testing levels presented in a typical V-model [19] which are performed to verify only software, the levels of CPS testing have to verify software, hardware, network and the integration of all these components to work as a single system.

In the V-model, unit testing represents code level or unit level testing, e.g. a single program module. It verifies the smallest functional code when isolated from the rest of the codes. For CPS, as shown in Fig. 2, the unit testing is performed at the hardware and software levels separately, to independently test the functionality of both hardware and software. Further, the network testing is important to verify the network operations and communication flow, which is not an integral/separate part of the testing levels in the V-model.

Integration testing verifies that different independent units can be integrated together and that they communicate correctly. Thus, the network testing is performed at both, unit and integration testing levels. In CPS testing it covers a bigger umbrella by encapsulating the integration of software components as well as the integration of hardware components, as compared to the traditional integration testing that refers to the integration of software components only.

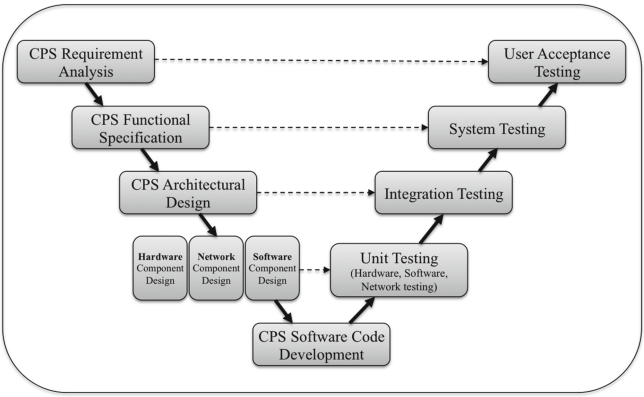

Fig. 2. Life cycle model for CPS

System testing verifies the whole application for its functionality, interdependency and communication. In the V-model, it usually consist of only a software part while in CPS, it encapsulates hardware, software, and network parts. User Acceptance testing is performed in a user environment and verifies that system is ready to use and meets user's requirement, and is relevant for CPS and V-model testing.

4 State-of-the-Art Survey

This section categorizes the SOTA survey based on the testing types described in the preceding section.

The string that we have used to search articles in Google scholar is ((cyber physical system) OR (medical systems) OR (real-time systems) OR (robot software) OR (vehicle software) OR (hybrid control system) OR (embedded system)) AND ((testing) OR (validating) OR (evaluating)). Based on the search string, we found totally 59 papers. We considered the most relevant and recent papers w.r.t. testing, and short-listed 16 papers, which are summarised in Table 1 and are explained below.

We see from Table 1 that most considered studies focus on System testing. There are a few studies on Hardware and Network testing. Further, we see that only *Real-time hybrid structural testing* [10,25] performs all six levels of CPS testing.

4.1 Hardware Testing

L. C. Silva, et al., present a model-based architecture for validating/verifying the Medical CPS [22]. Their architecture includes components representing models for medical devices and a model for the patient. The architecture can help developers/testers to generate test cases by validating these models.

Table 1. Relation between state of the art and testing types

CPS Testing Level \ State of the Art	[1]	[2]	[3]	[6]	[7]	[9]	[10]	[12]	[13]	[15]	[17]	[18]	[22]	[23]	[25]	[27]	Total
Hardware testing							✓				✓		✓		✓	✓	5
Structural testing			✓				✓		✓	✓	✓	✓	✓	✓	✓	✓	10
EFP testing			✓	✓	✓		✓		✓			✓			✓	✓	8
Network testing		✓				✓	✓				✓				✓		5
Integration testing		✓					✓				✓				✓		4
System testing	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓		✓	✓	12

Zhang, et al., address the challenges of generating test cases for CPS from Formal models [27]. They generate test cases from formal specifications by applying differential dynamic logic (DL). Differential DL is a logic for specifying and verifying hybrid systems. They translate the test cases, which are generated from formal specification into a model in Modelica language. Modelica is sometimes called a hardware description language that allows the user to specify mathematical models of complex physical systems, thus this testing approach can be applied to perform hardware testing.

Lim, et al. propose a hierarchical test model and automated test framework for robot software components of Robot Technology Component (RTC), which was combined with hardware module [17]. The proposed hierarchical testing procedure model included three levels of testing: unit testing, integration testing and system testing. Here the unit testing corresponds to the hardware testing. Hardware module is considered as a basic unit for hierarchical testing for robotic software component.

Real-time hybrid structural testing [10,25] is identified as a grand challenge for CPS and includes all six levels of testing. The physical test specimen in *Real-time hybrid structural testing* of Huang et al.'s experimental setup is composed of a small steel compression spring, which is used to represent the bending stiffness of an actual column in a portal frame structure, thus performing hardware tests [10]. Tidwell et al. present an initial work on a Cyber-physical Instrument for Real-time hybrid Structural Testing (CIRST). It targets to provide a highly configurable architecture for integrating computers and physical components, thus performing hardware tests [25].

4.2 Structural and Computation Testing

Conformance test using timed automata is another famous approach to perform structural testing and to test properties of the real-time system [6,13,18]. In this method, a conformance test algorithm is provided which constructs a set of test cases. The final output of the test method is either a Yes if the implementation conforms to its specification, or No when the implementation fails to conform to the specification because it fails a particular experiment.

Zhang, et al., generate test cases for CPS from Formal models by applying differential dynamic logic (DL) for testing in [27].

Badban, et al., present algorithms and techniques for automated test case and test data generation to test hybrid controlled cyber-physical systems [3].

A model-based architecture is used to test the Medical CPS [22]. A controlled experiment is performed to verify the behavior of components designed for the proposed architecture. The experiment analyzes the interaction among components. The most important concern in the Medical system is the patient's safety. Thus, the architecture focuses on the aspect that developers can test their applications without putting any risk on compromising the security of patients. Moreover, the data privacy is maintained, while keeping the generated data statistically compatible with real data.

Lee, et al., present embedded system software testing using a Service Oriented Architecture (SOA) method [15]. They present a mobile service testing process using test case specification. They conclude that service interoperability test process can extend the application testing to develop cost efficient and optimized mobile services. They analyze mobile application requirement, write service specification, optimize the design, and provide extended use case specification.

Srivastava and Kim develop variable length genetic algorithms to optimize software testing [23]. Genetic search algorithms are used to find critical path clusters in software code, and consequently, based on their identification they present a technique for optimizing testing and report preliminary results. Exhaustive software testing is intractable for even medium sized software. Therefore, they present a more selective approach to testing by focusing on those parts that are most error-prone and critical so that these paths can be tested first. The efficiency of testing is increased as their technique focuses on the most critical paths. Additionally, authors address that by applying generic algorithms they made an undependable technique from any specific problem and it can be of tremendous importance for users.

Lim, et al. perform structural testing within their hierarchical test model and automated test framework for robotics software by testing a series of operations for software component, which were specified in the document of software component requirement [17].

Structural testing is an integral part to test *real-time hybrid structural testing* [10,25]. Huang et al. perform structural testing using a case study of several fundamental interlocking challenges in developing and evaluating CPS [10].

4.3 EFP Testing

Goodloe and Pike check the real-time properties of distributed hard real-time systems using an online *monitor* [7]. They claim that testing and formal verification are not sufficient to demonstrate the reliability of real-time systems, and advocate online monitoring as a promising technique for making safety-critical real-time distributed systems more reliable. Online monitors execute as a separate process, check conformance to a specification or property at runtime and

can change the system direction into a known state if it deviates from its given specification, thus are better suited for safety-critical real-time system, like space shuttle and aircraft systems.

Badban, et al., present automated test case generation techniques that are applicable for testing the hybrid controlled embedded real-time systems like in avionics and railway [3].

As said in the previous section, *Conformance test* using timed automata is another approach to test properties of the real-time system. A new testing framework for real-time systems based on partially observable, non-deterministic timed automata and a formal conformance is proposed in [13]. The framework allows users to define the interface between the tester and the system under test (SUT) as well as assumptions on the environment of SUT via suitable model. The algorithms are provided in online or offline mode that generate analog clock or digital clock tests. Authors present coverage criteria to reduce the number of generated tests. The system is validated using a prototype test generation tool and two different case studies. Another study on automated derivation of functional test cases for real-time systems is proposed in [18]. A method for semiautomatic derivation of test cases is presented using formal specifications coded in TRIO language in order to fill a critical gap in the field of rigorous verification. The method is applied to several real-life case studies with industrial partners. The method discovers subtle errors that remained uncovered by human inspection and by using more-traditional techniques. Finally, authors also compare TRIO with the TRIO+ language. Another work based on the conformance test method using networks of timed automata is presented in [6]. Based on a testable model for real-time timed transition systems (TTTS) the author introduces fault hypotheses and a conformance test generation algorithm that constructs a set of test cases from a TTTS. The test view detects information on a particular set of tests, such as: the selection of relevant events to be observed, the mapping between implementation and specification events, the granularity of the observer's clock, a partition of test events into inputs and outputs. After selecting different test views, the tester can control the number of tests required: more detailed tests can be used for critical test purposes, and less detailed tests elsewhere. The authors claim that this method can reduce the number of test cases and can produce the effective use of features such as persistent variables. Moreover, testers can use independent test methods to check that assumptions made are reasonable and that users can easily define have their experiments in Uppaal specifications and different test views. Since complete test suites are expensive, authors believe this method is a proper way to reduce the cost when just critical parts of the system are tested by complete conformance test methods.

Zhang, et al., generate test cases from formal models. Modelica language is also used for the purpose of computer simulation of dynamic systems where behavior evolves as a function of time, thus it can be used to test timing properties [27].

Real-time systems are always tested extensively for the timing behaviour, like deadline misses etc. Thus, EFP testing is covered in *real-time hybrid structural testing* [10,25]. To examine the source of the deadline misses, Huang et al. measure the time for reading sensor data, writing actuator commands and other numerical computations [10]. Tidwell et al. claim that their testing method has broad impacts on both civil engineering and real-time computing and that it can enable real-time testing of a wide range of civil infrastructures and provide a CPS for the study and evaluation of real-time middleware [25].

4.4 Network Testing

A penetration testing policy is used by F. Alisherov and F. Sattarova [2] in order to test services and bring conformity between penetration testers and clients of the penetration test. It is a traditional testing method in which a tester send data to and from a secure system and then analyze the security measures of the system using a packet sniffer. This testing method falls under network testing since the data is sent to and from the system.

Lim, et al. [17] propose an automated test framework for robot software components in which the wireless communication station connects test-bed system to the main PC of robot test engine. Since they propose the wireless communication station in their framework they may use some technique(s) to test the communication. However this is not explained completely in the paper.

Real-time hybrid structural testing includes network testing [10,25]. Structural testing is typically based on data-flow, therefore, researchers explore the lower level details on testing the communication between ports (over network, local or shared memory).

He, et al. propose a cyber physical test bed (CPT) in order to visualize the environment of the wireless access and localize the body sensor networks (BSN) [9]. They design an analog channel emulator for Ultra Wide Brand (UWB) technologies. In order to verify their approach and to showcase the application of the CPT they accomplish some case studies. They evaluate the performance of data transmission inside the human body and TOA-based indoor localization. The results of case studies reveal the best performance of the indoor tracking system in the non-multipath condition. They also find the influence of the wooden wall and the metallic chamber in data transmission.

4.5 Integration Testing

F. Alisherov and F. Sattarova tested the integration and security of a CPS in [2] using penetration testing technique.

The second level of hierarchical testing procedure model is integration testing which is performed after unit testing for validation of hardware module [17]. The interoperability of hardware modules and software components was checked by performing integration testing while the robot hardware API was tested for performance index of functionality by using test cases.

Testing the integration of components in big distributed real-time CPS is very important and performed for *Real-time hybrid structural testing* [10,25]. Huang et al. test some of the components alone and visualize other components' behavior. They also perform integration testing by checking the synchronization among components [10]. Tidwell et al. perform integration testing using CRIST to test highly configurable architecture for integrating computers and physical components and provide a system for supporting real-time operations in distributed hybrid testing [25].

4.6 System Testing

Kane, et al., use system testing on a prototype vehicle design using a passive external runtime monitor to detect violations of high-level critical properties [12]. They limit the scope of testing in two respects: (1) by describing and testing only critical properties instead of complete behavior of system and (2) by providing approximate bounds to safety instead of specifying exact safety invariants. The method is applied on an automotive domain. Automotive networks periodically broadcast system state message. The simulation-based monitor reads the log file generated by the vehicle's CAN broadcast network, and verifies whether the execution trace satisfies the targeted properties or not.

Goodloe and Pike focus on testing system-level properties of a distributed real-time systems using online monitors [7]. *Conformance test* method [6,13,18] is performed on the complete integrated system, thus we categorize this method under system testing. Testing a Medical CPS includes detailed testing of hardware, structural and computations, and of complete system to ensure the patient's safety.

Modelica is a physical modeling language that allows tools to generate efficient simulation code automatically to facilitate exchange of models, and simulation specifications to test a simulation of complete cyber-physical system [27].

A classification of the concurrency bugs based on the observable properties is presented by Abbaspur et al. [1]. They categorized concurrency bug properties for concurrent and multicore application, that will help in system testing.

As described previously, the penetration testing is used to test the system's security measures [2], thus it falls under the system testing category.

The third level of hierarchical testing procedure model is system testing [17]. In their approach, the test cases are derived using black box testing techniques. The performance index of functionality includes completeness of function realization, correctness of data, compatibility of data and etc. Testing techniques of boundary value analysis, equal partitioning testing and state transition testing are used for system testing of robot software component.

The complete system is tested in real or using simulations for *Real-time hybrid structural testing* [10,25].

5 Challenges in Testing the CPS

The confluence of cyber and physical spaces of CPS technologies leads to new opportunities and subsequently some challenges in testing the system. Some of these challenges are summarized as follows:

- One of the essential challenges in validation, verification and certification of CPS is the current gap between formal methods and testing [20]. Thus compositional verification and testing methods that explore the heterogeneous nature of CPS models are necessary.
- Multiple parts of a CPS are often embedded systems with a real-time software executing. A real-time system requires to guarantee both (1) function correctness and (2) non-functional or extra -functional correctness. Extra-functional properties are closely related to the inherent interaction with the system environment. Examples of such properties may be temperature, power consumption, and timing [21,26]. Testing both functional and specially non-functional correctness in such systems is a challenge.
- The assertion of the correctness in designing and implementing CPS is another challenges. Correctness does not only encompass algorithmic and functional aspects but also extra-functional properties that are closely related to the inherent interaction with the system environment.
- Creating an automated or semi-automated method to evaluate the results of system testing is a limitation in CPS testing.
- To test run-time monitoring of real systems, the abstract model technique is used. In this technique, the real system is mapped to an abstract model. The runtime state information to perform monitoring is provided by this abstract model. However, abstract models will not be sufficient and it makes challenges on real system testing time.
- Defining the boundaries of the test landscape by environment is a challenge in testing the CPS. For instance, capturing physical limitations of devices or bounding the frequency of periodic inputs. Thus, system usually tested for the inputs specified, while it is necessary to also test for any inputs outside those ranges.
- The CPS which are related to electric power grid testing for power consumption has specific challenge since it requires ingrained measurements.
- The nature of some CPS are related to real-time, therefore testing the system with multiple time scales of interacting, distributed and control might be a challenge.
- Many CPS will highly interface with users. These CPS must be user-friendly to their many non-technical users. There are challenges in designing and testing these systems. For instance, testing a distance system for daily medical checks with couple of sensors and controls for old people or people with disabilities is a challenging task. Thus, testing the user-friendly property of this type of systems is a challenge.
- From real-time systems perspective the CPS research not only encounters the adaptation of components technologies and network systems, but also concerns using physical and logical properties (like physical laws, safety, security,

robustness, verification, energy and resource) [21]. Accordingly, compositional verification and testing methods should be adapted for the heterogeneous CPS models. To make a secure CPS with mentioned properties more work is required towards developing new verification and validation techniques. One approach could be bridge formal methods and testing approaches [20].

6 Discussion and Conclusion

This paper surveyed a number of past and recent efforts in CPS testing and verification and/or validation method. Several research groups have had ongoing efforts in the area for over a decade and have produced impressive tools. Consequently, there is a solid foundation of research in complete testing which can cover all phase of testing lifecycle of an application. Yet, little research to date has focused on testing and validation for CPS within health care, smart transportation, power grids and safety support, where suitable testing method can arguably have profound impact in preventing costly and possibly fatal system failures. Compositional verification and testing methods should be adapted for the heterogeneous CPS models. To make a secure CPS, more work is required towards developing new verification and validation techniques specifically targeting security vulnerabilities. Also, the use of formal methods in combination with testing, e.g. in the form of mode-based testing, has potential and should be further explored. Moreover, in the development of new CPS new mathematical foundation will be defined, therefore a variety of questions need to be resolved at different phases of software testing to trigger and ease the integration of the physical and cyber worlds.

Creating an automated or semi-automated method to evaluate the results of system testing is a challenge in CPS testing that also deserves attention.

Acknowledgments. This research is supported by Swedish Foundation for Strategic Research (SSF) via the SYNOPSIS Project.

References

1. Abbaspour A.S., Hansson, H., Sundmark, D., Eldh, S.: Towards classification of concurrency bugs based on observable properties. In: Workshop on Complex faUlts and Failures in LargE Software Systems (COUFLESS) (2015)
2. Alisherov, F., Sattarova, F.: Methodology for penetration testing. Int. J. Grid Distrib. Comput. **2**(2), 43–50 (2009). Citeseer
3. Badban, B., Fränzle, M., Peleska, J., Teige, T.: Test automation for hybrid systems. In: Workshop on Software Quality Assurance, pp. 14–21, ACM (2006)
4. Bertolino, A.: Software testing research: achievements, challenges, dreams. In: Future of Software Engineering, pp. 85–103 (2007)
5. Bozkurt, M., Harman, M., Hassoun, Y.: Testing web services: a survey. Department of Computer Science, King's College London, Technical report TR-10-01 (2010)
6. Cardell-Oliver, R.: Conformance tests for real-time systems with timed automata specifications. Form. Asp. Comput. **12**(5), 350–371 (2000)

7. Goodloe, A.E., Pike, L.: Monitoring distributed real-time systems: a survey and future directions. Technical report, NASA Langley Research Center (2010)
8. Gupta, S.K.S., Mukherjee, T., Varsamopoulos, G., Banerjee, A.: Research directions in energy-sustainable cyberphysical systems. Sustain. Comput. Inf. Syst. **1**(1), 57–74 (2011)
9. He, J., Geng, Y., Wan, Y., Li, S., Pahlavan, K.: A cyber physical test-bed for virtualization of RF access environment for body sensor network. IEEE Sens. J. **13**(10), 3826–3836 (2013)
10. Huang, H.M., Tidwell, T., Gill, C., Lu, C., Gao, X., Dyke, S.: Cyber-physical systems for real-time hybrid structural testing: a case study. In: 1st International Conference on Cyber-Physical Systems, pp. 69–78, ACM (2010)
11. Incki, K., Ar, I., Sözer, H.: A survey of software testing in the cloud. In: Software Security and Reliability Companion (SERE-C), pp. 18–23 (2012)
12. Kane, A., Fuhrman, T., Koopman, P.: Monitor based oracles for cyber-physical system testing. In: Dependable Systems and Networks (2014)
13. Krichen, M., Tripakis, S.: Conformance testing for real-time systems. Form. Methods Syst. Des. **34**(3), 238–304 (2009)
14. Lee, J., Kang, S., Lee, D.: A survey on software product line testing. In: Proceedings of the 16th International Software Product Line Conference, vol. 1, pp. 31–40. SPLC 2012, ACM, New York, NY, USA (2012)
15. Lee, M.H., Yoo, C.J., Jang, O.B.: Embedded system software testing based on SOA for mobile service. J. Adv. Sci. Technol. **1**(1), 55–64 (2008)
16. Li, Y.F., Das, P.K., Dowe, D.L.: Two decades of web application testing–a survey of recent advances. Inf. Syst. **43**, 20–54 (2014)
17. Lim, J.H., Song, S.H., Son, J.R., Kuc, T.Y., Park, H.S., Kim, H.S.: An automated test method for robot platform and its components. Int. J. Softw. Eng. Its Appl. **4**(3), 9–18 (2010)
18. Mandrioli, D., Morasca, S., Morzenti, A.: Generating test cases for real-time systems from logic specifications. Trans. Comput. Syst. **13**(4), 365–398 (1995)
19. Mathur, S., Malik, S.: Advancements in the V-Model. Int. J. Comput. Appl. **1**(12), 29–34 (2010)
20. Rajkumar, R.R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: 47th Design Automation Conference, pp. 731–736 (2010)
21. Sha, L., Gopalakrishnan, S., Liu, X., Wang, Q.: Cyber-physical systems: a new frontier. In: IEEE International Conference on Sensor Networks, Ubiquitous and Trustworthy Computing 2008, SUTC 2008, pp. 1–9 (2008)
22. Silva, L.C., Perkusich, M., Bublitz, F.M., Almeida, H.O., Perkusich, A.: A model-based architecture for testing medical cyber-physical systems. In: 29th Annual ACM Symposium on Applied Computing, pp. 25–30, New York, NY, USA (2014)
23. Srivastava, P.R., Kim, T-h: Application of genetic algorithm in software testing. J. Softw. Eng. Its Appl. **3**(4), 87–96 (2009)
24. Tevanlinna, A., Taina, J., Kauppinen, R.: Product family testing: a survey. SIGSOFT Softw. Eng. Notes **29**(2), 12–12 (2004)
25. Tidwell, T., Gao, X., Huang, H.M., Lu, C., Dyke, S., Gill, C.: Towards configurable real-time hybrid structural testing: a cyber-physical system approach. In: ISORC, pp. 37–44 (2009)
26. Woehrle, M., Lampka, K., Thiele, L.: Conformance testing for cyber-physical systems. ACM Trans. Embed. Comput. Syst. **11**(4), 84 (2012)
27. Zhang, L., He, J., Yu, W.: Test case generation from formal models of cyber physical system. J. Hybrid Inf. Technol. **6**(3), 15 (2013)