

Review

A Comprehensive Technological Survey on the Dependable Self-management CPS: from self-adaptive architecture to self-management strategies

Peng Zhou^{1,2}, Decheng Zuo¹, Kun Mean HOU², Zhan Zhang¹, Jian Dong¹, Jianjin Li², Haiying Zhou¹

¹ School of computer science and technology, Harbin Institute of Technology, Harbin, Heilongjiang, China; zhoupeng@ftcl.hit.edu.cn, zuodc@hit.edu.cn, zz@ftcl.hit.edu.cn, dan@hit.edu.cn

² LIMOS, UMR 6158 CNRS, University Clermont Auvergne, Aubière CEDEX, France; kunmean.hou@gmail.com, jjli@isima.fr

³ School of electrical and information engineering, Hubei University of Automotive Technology, Hubei, China; zhohuhy_dy@huat.edu.cn

* Correspondence: zhoupeng@ftcl.hit.edu.cn, zz@ftcl.hit.edu.cn, Tel.: +86-0451-86403316

Abstract: Cyber Physical Systems (CPS) has been a popular research area in the last decade. The dependability of CPS is still a critical issue, and rare survey has been published in this domain. CPS is a dynamic complex system, which involves various multidisciplinary technologies. To avoid human error and to simplify management, self-management CPS (SCPS) is a wise choice. And to achieve dependable self-management, systematic solution is necessary to verify the design and to guarantee the safety of self-adaptation decision, as well as to maintain the health of SCPS. This survey first recalls the concepts of dependability, and proposes a generic environment-in-loop processing flow of self-management CPS, and then analyzes the error sources and challenges of self-management through the formal feedback flow. Focus on reducing the complexity, we first survey the self-adaptive architecture approaches and applied dependability means; then we introduce a hybrid multi-role self-adaptive architecture, and discuss the supporting technologies for dependable self-management at the architecture level. Focus on dependable environment-centered adaption, we investigate the verification and validation (V&V) methods for making safe self-adaptation decision and the solutions for processing decision dependably. For system-centered adaption, the comprehensive self-healing methods are summarized. Finally, we analyze the missing pieces of the technology puzzle and the future directions. In this survey, the technical trends for dependable CPS design and maintenance are discussed, an all-in-one solution is proposed to integrate these technologies and build a dependable organic SCPS. To the best of our knowledge, this is the first comprehensive survey on dependable SCPS building and evaluation.

Keywords: Cyber Physical Systems; Industry 4.0; MDE; Lifetime Verification & Validation; Dependability; Correctness; Flexibility; Real-time self-adaptation, Self-management; Self-healing;

1. Introduction

Cyber Physical System (CPS) was first proposed by the American National Science Foundation (NSF) in 2006. One year later, the President's Council of Advisors on Science and Technology (PCAST) raised the CPS to ensure the continued leadership of the USA and recommended to put CPS as a top research agenda. In 2012, German government put forward **Industry 4.0** to develop Germany's competitive position in manufacturing. Thereafter, EU (Horizon 2020, in 2013), Japan (CPS Task Force, in 2015), and China (Made in China 2025, in 2015) published their own plans on

CPS/Industry 4.0. Roughly speaking, CPS stems from the information and communications technology, and Industry 4.0 is based on the automation manufacturing technology. They are two different evolutions for the same goal [1]. CPS integrates computation, networking, and physical dynamics [2-3], it has even been regarded as a next revolution of technology, which can rival the contribution of the Internet [4].

Besides CPS and Industry 4.0, lots of similar concepts have been proposed to describe the system from different perspectives [5-7]. Here, we just introduce the most similar technical concepts. From the view of architecture, there are Machine to Machine/Man (**M2M**) [8] and System of System (**SoS**) [9]. From the view of networked control, there are Wireless Sensor Actor/Actuator Networks (**WSAN**) [10] and Networked Control Systems (**NCS**) [11]. From the view of communication and information processing, there are Internet of Thing/Web (**IoT/IoW**) [12], Fog [7, 13], Big data, self-adaptive control system, etc. In this paper, we use the term “CPS” to denote such smart closed-loop systems.

With the rapidly increasing complexity, it is impossible for administrators to understand the complex massive data and then give proper commands in time to instruct CPS to take right activities. Thus, self-management (or autonomous computing) seems to be the only alternative solution. Some exploratory researches have already been proposed to improve some qualities of self-management, such as reconfigurability [14] and interoperability [15] and the QoS of services [16]. The concepts of self-* and autonomic computing have been proposed for a long time [17-18]. Many models and strategies have been proposed for building self-adaptive systems [19-20]. However, due to limitation of current technologies, self-adaptation decisions are generally short-sighted, ineffective and unconstrained, which can't well satisfy the safety-critical requirements. Thus, it is still an open issue to build highly dependable, controllable and predictable CPS.

CPS involves numerous multidisciplinary technologies. For example, smart manufacturing involves Smart Factory, IoT, Smart Sensor, Big Data, Additive Manufacturing, and Hologram, etc. [6, 21]. However, these technologies are studied separately, few researchers discuss the interrelationships and integration of them in detail. Massive challenges still need to be overcome to build an organic, dependable SCPS with these fragmented technologies [22-23]. Among these challenges, guaranteeing the dependability of self-management is an urgent issue. Systematic solution is necessary to develop a dependable self-management CPS (SCPS), which includes dependable and flexible architecture design, creatively integration of current technologies, and strictly dependability verification. Moreover, a dependable SCPS should be able automatically to evaluate the dependability and the risk of the self-management strategies at run-time.

According to the goals of adaption, the self-adaption of SCPS can be classified into two types. One is *environment-centered adaption (self-adaptation)*, whose target is external systems (e.g. physical world and humans); it is aimed at interacting with the changeable environment properly. Another is *system-centered adaption (self-healing)*, which is aimed at guaranteeing dependability of the cyber space (e.g. the reliability of infrastructures and the availability/quality of services). For environment-centered adaption, SCPS should guarantee the fitness/correctness of self-adaption decisions, and the dependability (safety) of the adaptation procedures. For system-centered adaption, SCPS should automatically diagnose the faults, remove or isolate the failures, and adjust its structure and behavior to keep the system healthy. In the real world SCPS, **these two kinds of adaptation interfere with each other, and even conflict with each other in some cases** (e.g. resource competition, and the tradeoff between energy budget and redundancy). It needs great wisdom to design a dependable SCPS, which can maintain the dependability by itself and smartly balance the dependability with other requirements in different scenarios.

In this paper, we use the term “self-adaptation” to represent the environment-centered adaption, and the dependability of self-adaptation is mainly about the correctness and safety of decisions. And we use the term “self-healing” to represent the system-centered adaption, which focuses on improving the dependability of the platform of CPS. “Self-management/

self-management" is an integrated concept which includes "self-adaptation", "self-healing" and more other concepts, such as self-protecting and self-optimization.

1.1 Motivation and goal of this survey

It can never emphasize too much the importance of the dependability to a safety critical CPS. Because **1)** with the increasing complexity of system, it becomes more and more difficult to evaluate the dependability at design period. **2)** CPS has to continuously and autonomously adapt to the changeable environment in almost real-time, which introduces serious challenges to runtime dependability V&V. **3)** The effects on physical space can be hardly eliminated. CPS should carefully evaluate the risk of the decisions and guarantee the safety of the decision processing. **4)** Whether subsystems fail or not, activities should be started at the right time in the right place, and processed at the proper speed. Hence, it urgently needs systematic solution to guarantee the dependability of CPS at both design period and runtime. Otherwise, CPS would not simplify our life but make the life tougher and more dangerous. A comprehensive technological survey is needed to guide the further research and improve the dependability of CPS.

There are 7948 papers are published on the topic of "CPS or Industry 4.0" in more than 115 fields according to the "web of Science" due to 2018-8-30 (the detailed results are shown in the Figure 14 and Figure 15 in the Appendix A). However, the publications on the dependability of CPS is few. Only 456 papers belong to the topic of "dependability or reliability or availability" (detailed result is shown in Figure 16 and the Table 4 in the Appendix A). With further review, only 67 papers discuss the dependability in one or more sections. A similar conclusion is made in the survey on component-based CPS architecting since 2015 [24], where only 9 of 1103 publications discuss the reliability, and 6 papers are about maintainability. Currently, most surveys just mention the importance of dependability to CPS, only one survey focus on the challenges of dependable infrastructures of CPS and discusses the dependability of CPS in detail [25]. With more and more researcher pay attention on dependability (as seen in Figure 16), it urgently needs one comprehensive survey on the dependability of CPS to guide the future research

1.2 Literature Search rule

In this survey, we focus on the dependability of CPS and searched the papers with 4 group keywords on *Web of Science*, *ACM Digital Library*, *IEEE Xplore*, *Springer Digital Library*, *Elsevier Science Direct* and *Wiley Online Library*. First group keywords are the similar concepts of "CPS", the priority of terms is illustrated in follows: "Cyber physical System" = "Industry 4.0" > "Internet/Web of things"(IoT/WoT) = "Fog computing" > "System of System" (SoS) > "Machine to Machine"(M2M) > "Wireless sensor network"(WSN) = "Wireless sensor actuator network" (WSAN) > "networked control system" > "embedded system", which are denoted by *key_A*. And the second group keywords *key_B* are about the self-* characteristics of CPS, which include "self-adaptation", "self-adaptive", "self-adapting", "self-healing", "self-maintaining", "self-management" and "autonomic computing". The third group keywords *key_C* are dependability related, which include "dependable/dependability", "reliable/reliability", "maintainability", "fault-tolerance", "safety", "fault detection", "fault diagnose/fault diagnosis", "fault prediction" and "fault prevention". We searched the papers with the combination of "*key_A and key_B and key_C*". Meanwhile, SCPS can also be regarded as a large scale real-time self-adaptive system. We also used "*real-time and key_B*" as the fourth group keywords to investigate the self-management strategies. Moreover, we searched the papers on dependable self-management strategies with the combination of "*real-time and key_B and key_C*". We eliminated the duplication and checked the abstract to select most relevant papers. Finally, we cited 240 papers and 9 books among 1376 papers.

Notice that, both dependability management and SCPS development are systematic engineering which integrate massive technologies. To avoid bottleneck effect, it needs systematic architecture design and elaborate integration of technologies. Generally, such approaches need a

long-form introduction and comprehensive analysis. Hence, we prefer to cite the *full paper* rather than the short paper. To cover the advanced self-management strategies, more conference papers are referenced. In addition, we prefer to cite the researches on theory and practice rather than system introduction, and the papers on systemic solutions rather than concepts, if they share the same topic/domain. ESI/SCI paper and highly cited paper are cited first.

1.3 Structure of this survey

To build a dependable SCPS, it is necessary to co-design and co-evaluate the applied architecture and strategies. This work tries to provide a detailed, comprehensive investigation on dependability engineering and life cycle maintenance for SCPS. As SCPS is built with the existing (embedded) systems, to avoid unnecessary duplication of work, we assume that readers have a good background in the dependability, control system and embedded system. In this survey, we will focus on the new technological challenges of SCPS modeling, dependability verification & validation (V&V) and runtime dependability management. To simplify, we regard the subsystem as the unit and focus on the dependable integration among subsystems, and the dependable interaction between the cyber space and the physical space. This survey will discuss at three levels: **1)** the self-adaptive architecture (including architecture and model based V&V), **2)** dependable environment-centered adaption (self-adaption), and **3)** system-centered adaption (self-healing).

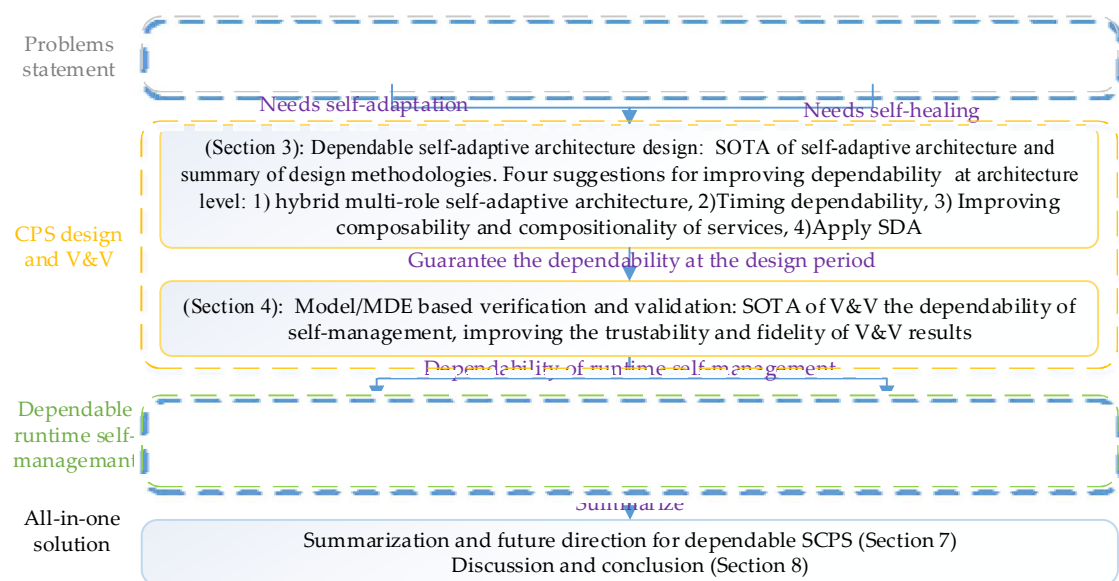


Figure 1. Structure and relationship of survey

The rest of the paper is organized in an order of SCPS engineering (dependability requirement analysis, SCPS architecture design and Model based V&V, and system maintenance), as illustrated in Figure 1. It first recalls the concepts of dependable SCPS, and then proposes generic processing flow, error sources and challenges of dependable SCPS engineering and run-time management in Section 2. We survey the state of the art in self-adaptive architecture of SCPS, summarize the shortages of current architectures, and then propose a conceptual dependable self-adaptive architecture and the dependent technologies in Section 3. We split self-management solutions into two sections, and discuss the self-adaptation (environment-centered adaption) technologies in Section 5 and self-healing (self-centered adaption) technologies in Section 6. In Section 7, we complete the jigsaw of technologies and discuss the technical trends of dependable SCPS. An all-in-one conceptual solution is proposed for future SCPS development and maintenance. In section 8, some interesting features of SCPS and issues beyond the technical area are discussed; we also conclude the solutions for 9 challenges which proposed in section 2.

2. Background and overview of challenges of dependable SCPS

The increasing complexity is a fundamental challenge to both SCPS design and maintenance. To design a dependable SCPS, we need efficient model theory and high fidelity model based engineering (MDE) toolset. And to simplify maintenance, we need a self-adaptive architecture, dependable self-adaptation strategies and self-healing solutions. In this section, we first recall the concept of dependability and the capability of dependable SCPS. Then we introduce the generic self-management flow of dependable SCPS. Next, we analyze the error sources of SCPS with a formal self-adaptation flow. Last, we introduce the role that the feedback mechanism plays in the self-healing solutions. Finally, we present the challenges of CPS engineering (modeling and V&V).

2.1 The methodology for dependable SCPS engineering

Dependability is an integrating property, which encompasses three concepts: 1) the threats, which consist of errors, faults and failures; 2) the attributes, which include reliability, availability, safety, confidentiality, integrity, and maintainability; 3) the means to achieve the dependability, which contain fault prevention, fault tolerance, fault remove, fault forecasting. More detailed introduction on dependability and its threats and attributes refer to [26]. A dependable system should be able to notify the administrators of the risk of permanent faults, help administrators to identify the sources of errors and locate the failed subsystems. Moreover, the dependable system should be able to recover from the transient faults, isolate and tolerate the permanent faults.

However, CPS is so complex that no administrator can clearly understand its behavior and identify the symptoms in time, let alone make proper adaptation decisions and maintenance decisions. To simplify management, one promising solution is **applying self-management solutions (a.k.a. autonomic computing) to tame the complexity of management**. Self-management can be subdivided into *self-adaptation*, *self-healing*, *self-configuration*, *self-organizing*, *self-synchronization*, *self-protection*, *self-learning* and *self-optimization* [19-20, 27-28]. Through self-* solutions, SCPS can automatically deal with internal and external changes, and maintain the quality of services. To reduce the complexity introduced by self-management services, we should **systematically design the architecture and the interoperation interfaces, to facilitate self-management. Moreover, we should build a SCPS that can aware its statuses, model itself and the runtime requirements of self-management activities**. So that SCPS can guarantee the quality of self-management and the dependability of activities.

Building a dependable SCPS needs to comprehensively integrate various technologies at different levels. *At the infrastructure level*, CPS needs more reliable, stable hardware and software to tolerate the interference from natural environment. *At the subsystem level*, CPS should be able to diagnose or predict the failures automatically, because there are too many subsystems for supervisor to detect one by one. *At the service level*, we need to develop high available solutions to provide 24/7 service supporting. *For decision making*, CPS should first provide integral and consistent information for Decision Support System (DSS), and then evaluate the fitness (i.e. correctness and safety) of decision at run-time. *For decision process*, CPS should select the most reliable services, and guarantee the reliability of commands and the safety of the process of decision. *For daily maintenance*, we need to improve the flexibility, observability, traceability to simplify the failure location and manual recovery. Furthermore, considering the large potential value of data, it is necessary to guarantee the confidentiality of information in the CPS. In this survey, we mainly focus on reliability and availability of architectures for SCPS, safety and maintainability of self-management at run-time.

As errors are accumulated with the procedure of self-management loop, undependable self-management will lead to more serious damages. It is necessary to verify each step of the self-management loop, which implies that we need to develop additional services to monitor and verify the self-management services. As we all know, complexity negatively affects the dependability. Therefore, it is necessary to reduce the side-effect of introducing self-management and to simplify the daemon services of self-management. To tame the complexity, one key rule is

using simplicity to control complexity [29]. For SCPS, we should use relatively simple self-management services to control complex normal functions of CPS, and use simpler daemon to monitor self-management services. To achieve this goal, we need to find the universal schemas of CPS's behavior and the related self-management services. In case of improper decisions, we also need to improve the observability, traceability and maintainability of SCPS. Thus, the administrators can supervise the CPS, fine-tune the strategies and take over the system in emergencies to avoid catastrophes.

2.2 The process flow of dependable SCPS

CPS is a kind of closed-loop system with multi-actors, which include sensors, actuators, computing system (such as data collectors and analyzers, and decision support system), and communication networks, and even human beings [5,21]. Yet, few publications discussed the influence of physical space/world on the cyber space in detail. Without the physical space, the closed-loop of interaction is incomplete. *To interact dependably, SCPS should not only consider the behavior of cyber space, but also take into account the physical effects (on the cyber space), even the effects of human beings.* Imagine that a man drives a car in the terrible cold weather. The low temperature will not only slow down his reaction time, but also reduce the reliability of the car (i.e. reducing the friction between the tire and ground, weakening the strength of materials). A dependable self-driving system should aware the effects of cold weather and adjust the related weights to make proper decisions. For another example, after water crossing and wading, the braking effect becomes weak and the braking distance increases. A self-driving car should aware the change of braking distance and adjust the related parameters (such as the time to start braking). Furthermore, the dependable self-management should be perspicacious and far-sighted, so that the actuators can deal with the changes.

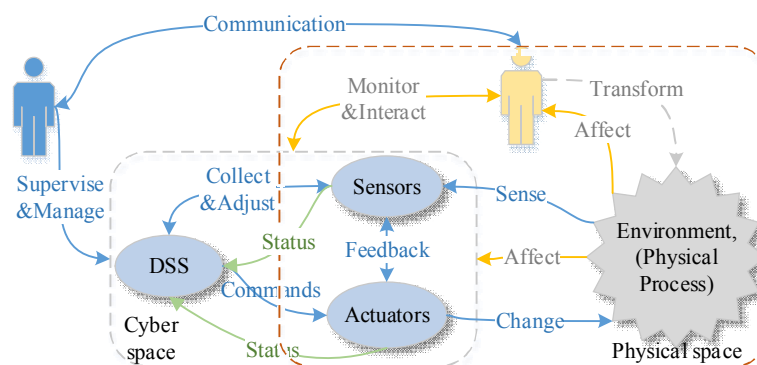


Figure 2. The generic process flow of environment-in-loop CPS

To adapt to the changeable environment, SCPS has to deal with uncertainties and should be able to tolerate unexpected failures at runtime. In detail, SCPS should be able to quickly sense the environment, aware its context, and predict the future, so that it can adjust its behavior to adapt to the environment or change the environment to protect itself (i.e. heating the battery in cold environment). Meanwhile, SCPS should form a complete, sensitive and soundness closed reaction loop to avoid over operation. From this point of view, we propose a generic processing flow of environment in-the-loop SCPS, which is illustrated in Figure 2. The core participators in cyber space include sensors (network), actuators (network), networks and DSS. Sensors and actuators are the interfaces between cyber space and physical space. The physical system includes the devices of sensors and actuators, the human beings and the physical environment around them. We don't take the DSS as a part of physical space because they are normally deployed in the ideal environment, which are barely affected by the physical/natural world. In this figure, the blue arrows represent the data flows of traditional CPS. The yellow arrows stand for the effects of the physical environment on

the human and the devices. And the green arrows show the flow of the status information about infrastructures (e.g. Sensor, actuator and network), which is very important to evaluate the dependability of SCPS. Dependable SCPS should monitor the statuses of physical systems (green arrows), and evaluate the effect of physical systems (yellow arrows), and guarantee the dependability of traditional process flow (blue arrows).

Notice that the status of infrastructures also include the physical information, i.e. temperature of board, RSI (Relative Strength Index) of communication channel. DSS should take the physical effects on the infrastructures into account during decision making. And for human-in-the-loop CPS, DSS even should consider the intentions of human beings, reserve enough response time when interacting with human beings. As there is few efficient theory to model the human behaviors now, we will just briefly discuss related researches and issues in section 7.

2.3 Formal processing flow of self-management and error sources

The formal self-adaptation flow is illustrated in Figure 3. $S_c(t)$ and $S_p(t)$ are the statuses of cyber space and physical space at time t respectively. $S_c^m(t)$ and $S_p^n(t)$ are the (minimum redundancy maximum relevance) observed statuses of cyber space and physical space, where $S_c^m(t) \subseteq S_c(t)$ and $S_p^n(t) \subseteq S_p(t)$. The inputs for DSS are the sequences $(S_c^m(t-j), \dots, S_c^m(t))$, $(S_p^n(t-i), \dots, S_p^n(t))$ and the periori knowledge. $\{D_c(t)\}$ is the self-healing decision and $\{D_p(t, p)\}$ is the self-adaptation decision. $\{\oplus \Delta S_c(t)\}$ is the self-healing activity of one subsystem and $\{\oplus \Delta S_c(t)\}$ is the cyber space statuses changed by all self-healing activities (a.k.a. the effects of self-healing activities) at time t . Likewise, $\oplus \Delta S_p(t)$ and $\{\oplus \Delta S_p(t)\}$ represent the self-adaptation activity and the changed statuses of physical space. $\oplus(\int_t^{t+1} f_p(t)dt + c)$ is the inertia effect of the physical phenomenon and \oplus is a non-linear operator. The statuses $S_c(t + \Delta t)$ and $S_p(t + \Delta t)$ can be changed by the self-management activities, the inertia of physical processes and human being etc.

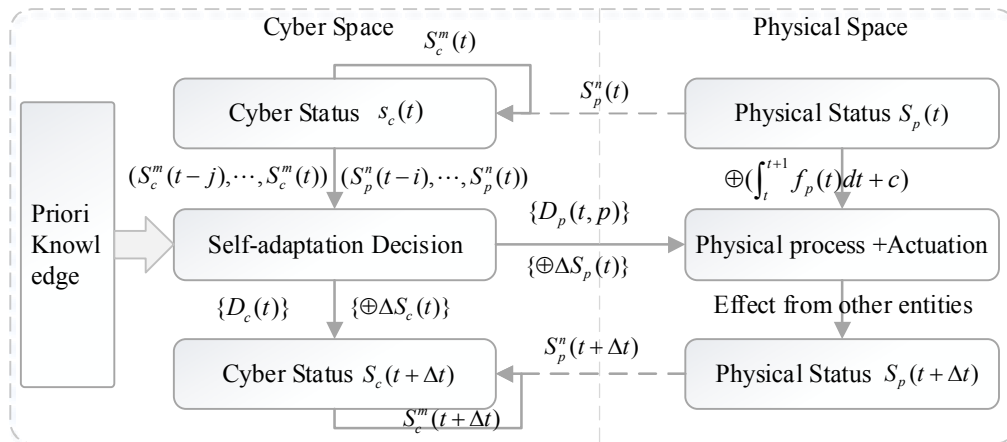


Figure 3. The formal interaction between cyber space and physical space

No matter we model the self-management loop as a linear control system $\dot{s} = A(t)s + B(t)u + n_e$ or a nonlinear control system $\dot{s} = f(s, t, u) + n_e$ (where $s = S_p^n \in \mathbb{R}^n$ or $s = S_c^m \in \mathbb{R}^m$, $u \in \mathbb{R}^k$, n_e is the error), we still suffer from some hard issues that caused by errors n_e . Generally, n_e is modeled as a random variable, but in the real world system, n_e depends on the error of observation values and the related operating functions, as well as their weights. Due to the limitation of current technologies, it is impossible to collect complete and accurate features, and it's also impossible to

analyze the abundant data in time. In practical engineering, we have to use $S_p^n(t)$ as the approximations of $S_p(t)$. Moreover, as the environment continuously changes, SCPS should dynamically adjust the weight matrices (A , B), the selected statuses S_c^m and S_p^n , their dimensions (n , m), and the length of the historical status sequence (i), as well as the control rules u . To well fit the scenarios and to achieve dependable self-management, additional high-level rules or priori knowledge are necessary to instruct the SCPS to choose the optimal parameters and control rules. However, no existing theory can identify the scenario and automatically search the corresponding optimal parameters in time. Even if we have developed such a theory, it is still difficult for SCPS to accurately eliminate the error n_e . Moreover, SCPS has to search the sub-optimal combination of functions (or parameters) and generate defective decisions in a limited time Δt (notice that, Δt maybe a variable value). Such a compromise also inevitably increases the risk of failures.

In a nutshell, the error sources (issues) of self-adaptation are as follows: 1) **long-term predictions are unreliable**, because the physical effects on the reliability of infrastructures are nonlinear (a.k.a. The Butterfly Effect). 2) **Statuses for decision making are incomplete**, $s_c^m(t) \subseteq s_c(t)$ and $s_p^n(t) \subseteq s_p(t)$. 3) **$s_c^m(t)$ and $s_p^n(t)$ are not trustable**, because the sensors may fail and data transmission may introduce errors. 4). **The timestamp of every sub-statuses (events) are not completely the same**. As the clocks of subsystems are not identical, we have to use the set $s_p^n(t) = \langle s_p^1(t^1), \dots, s_p^n(t^n) \rangle$, whose time is $t^1 \approx \dots \approx t^n \approx t$. This means that DSS may have different orders with original events, and the causal relationship of events may lose. 5) **Short-term decision may be invalid or easy to miss the deadline**. Plans always fall behind the changes, so the predicted future is not identical with the real future $s_p(t) \oplus \Delta s_p(t) \neq s_p(t + \Delta t)$. What's worse, and failures may have occurred during $[t, t + \Delta t]$. *The time maybe too short to find a self-healing solution, or even there may be no reasonable and practicable self-healing solution.* 6) **There is no effective theory to model the changeable factors (with the fixed matrixes $s_p^n(t)$ and $s_c^m(t)$)**. Generally, the selected KPI (Key Performance Indicator) factors and the weights of them are not fixed. Fast context-aware solutions are necessary to search the optimal set of factors and weights. Further survey on dependable real-time self-adaptation will be discussed in section 5.

2.4 The schemas of feedback loop for self-healing and self-reference problem

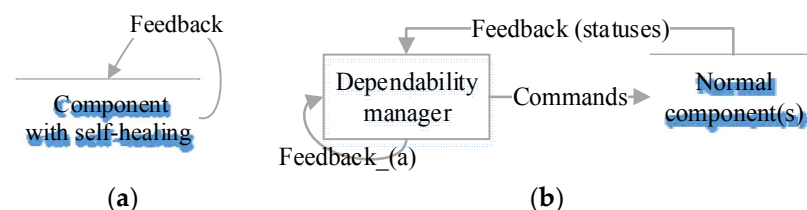


Figure 4. The schemas of self-healing (a) Internal self-healing in a signal component; (b) External self-healing with the schema of diagnose-prediction-prevention

Self-healing is an important capability of the dependable SCPS. Compared with the self-adaptation, self-healing suffers from not only the same issues caused by errors, but also the self-reference problem. Generally, the feedback schemas of the self-healing solutions can be classified into two types, which are illustrated in Figure 4. For schema (a), the self-healing measures are highly integrated with normal function, the component will adjust its behavior to improve dependability during executing, i.e. some dependable control mechanisms [30]. For schema (b), the self-healing measures and the normal functions perform independently in space-time, i.e. expert system based self-healing. Compared to schema (a), schema (b) has higher flexibility and scalability, but it may react more slowly. Generally, schema (b) is more efficient in the large scale SCPS, and the

causes will be explained in section 3.3.3. Notice that, the manager in schema (b) is a component that contains the schema (a).

According to the occasion of taking healing measures, self-healing methods can be classified into two types: *preventive healing* and *remedial healing*. The preventive healing methods monitor the symptoms, predict faults, and then take actions to prevent/avoid the failures. These methods can be regarded as a special self-adaptation services which are dedicated to the system itself. In this context, **preventive healing method suffers from all the issues that self-adaptation does** (as seen in subsection 5.2), **but even more serious** (because self-healing may also fail). Remedial healing tries to reprocess the services from some backup statuses after failures occur, and remove/isolate the error sources. As the recovery measures have to reprocess services, it increases the risk of deadline missing. Moreover, due to fault propagation, it is very difficult to locate and remove the error sources in some cases. Consequently, similar failures may occur repeatedly and reprocessing may fail again and again. What's worse, **self-healing suffers from the halting problem** [31] (which is a problem caused by self-reference, and it is an unsolvable logic puzzle). As seen in the schema (b), we also need to guarantee the dependability of the manager (because self-healing is also a service). For this purpose, we have to introduce a self-reference service (a.k.a. the service likes schema (a)), otherwise the self-healing chain is unclosed and some services can't be healed.

2.5 The challenges of guarantee the dependability of SCPS

Due to the increasing complexity, it is impossible to enumerate all scenarios and test all cases during the design period. What is worse, the traditional fault management strategies are dedicated to the particular organization in specified scenario, which are unable to effectively handle the failures caused by the dynamic behaviors. In this sub-section, we summarize three new challenges of (model based) dependability evaluation of self-adaptation and the relative solutions will be surveyed in section 4. We also propose six new challenges of runtime dependability management of SCPS, the available self-healing solutions will be investigated in section 5.

2.5.1 The legacy issues

Though dependability has been researched for years, several challenges are still unresolved: **1)** The increasing complexity makes the problem harder, i.e. Testing is NP-hard [32], which implies that the published system inevitably contains bugs. **2)** The cost of development increases exponentially with the degree of reliability [29]. **3)** Self-healing (self-detection) of Turing machine system is a halting problem [31], which is a famous paradox in computability theory. In short, it is impossible to design a generalized algorithm (over Turing machines) which can determinately find out whether the program halts (fail) or not. The halting problem implies that it is impossible to achieve absolutely dependability with self-healing solution. Hence, manual interventions are still necessary to maintain the dependability of SCPS. Nevertheless, self-healing can reduce the interventions of human being and simplify the manual management. Otherwise, managing a complex system like SCPS will be a disaster.

For SCPS, dependability related issues are tightly intertwined with correctness issues. Unreliable data may mislead SCPS to make wrong self-healing decisions, and wrong decisions will reduce the dependability or even cause terrible failures. As the traditional fault management strategies can't evaluate the correctness/fitness of decisions, these strategies are unable to effectively deal with faults caused by the misbehavior. Take the redundancy strategy as an example, all homogeneous sensors suffer from the same interference from the environment, the redundant sensors can't tolerate such failures. Likewise, the multi-copy of one same algorithm can't tolerate the inputs with errors and can't make good any defect of design. In contrast, the redundant homogeneous subsystems may have inconsistent observations, which will trigger different even conflicting activities.

Moreover, for traditional systems, their structures are predesigned and fixed, or rarely changed throughout the whole lifetime. Systems can only adjust their logical behavior, for example, selecting a new logical branch (i.e. the if-else pattern), switching the state of next operation (i.e. the feedback control $x(t+1) = Ax(t) + Bu(t)$), or renewing the threshold/reference value (i.e. time series models $x(t+1) = \sum_{i=0}^n w_i x(t-i)$). Systems barely change the organization or the topology of the (physical/logical) components. For example, the TMR (Triple Modular Redundancy) system can decrease the connection degree (between the majority gate and the redundant modules) from three to two and then to one. But all modules and links are predesigned, no new module or (physical/logical) link is appended in the degraded or recovered system. While to SCPS, new subsystems and links may be added and removed dynamically.

The dynamically changed structures of SCPS introduces another big challenge to system design and V&V. As we all know, it is impossible to verify all possible structures at the design period or at the plan making period. As a compromise, SCPS has to search the optimal topology and select the best candidates at runtime, and then reorganize them as a temporary but cohesive team. Furthermore, subsystems may quit and join the team at any time because of failures, recoveries or movements. For each time of re-organization, SCPS should evaluate the risk and dependability of structure changes. In other words, every subsystem should be able to model the context and evaluate the risk and dependability at run-time. To achieve these objects, subsystems should apply **model@run.time** methods to support **self-modeling** and **self-evaluation**. Moreover, all model@run.time methods should finished in a limited time.

2.5.2 Technical challenges of CPS modeling and dependability analysis

As the future is uncertain, SCPS has to model at runtime to reduce the uncertainties and generate several backup plans. Meanwhile, SCPS should check the status and properties of (heterogeneous) candidates, then select the most suitable implementers to optimize the plans. To achieve these goals, SCPS should model both the physical space and the cyber space, and predict the future environment and assign activities to the proper candidates. As well, SCPS should evaluate the failure risk and guarantee that the selected candidates can well satisfy the requirements of decisions. It needs efficient theories and complex engineering solutions to enable SCPS to automatically model the dynamic structures and the stochastic behaviors. It is also full of challenges to achieve optimal compromise between dependability and other requirements under uncertainties.

MQ1: Error sources and their failure distribution in different scenarios: It needs tremendous amount of investigation about the error sources and the relative failure distributions in different scenarios. What's worse, we don't have the accurate distribution of most failures at the design period. And at runtime, the distributions and their scale parameters may change with time, i.e. due to aging of hardware, the failure rate λ of exponential distribution increases with time.

MQ2: Fault propagation between subsystems in different modes (state space explosion and the validity of state combination): Each subsystem has several normal modes (i.e. high speed mode and energy saving mode) and error modes (i.e. degrade mode and failure mode). For different combinations, the paths and the effects of fault propagation are completely different. It is impossible to verify all possible combinations. Moreover, some modes are incompatible with each other, but we can't forbid the combinations of them by rules, because some combinations are invalid in some scenarios but are valid in others, and we can't distinguish these scenarios through rules or enumerations (at the design period).

MQ3: Modeling the dynamic structures and self-adaptation behaviors (triggered by multiple stochastic physical signs) and evaluate the failures caused by multi-errors: Most modeling methods are dedicated to static structures [33]. Few methods, especially few formal methods, support to model the dynamic structures and the stochastic interaction behaviors among subsystems. Hence, it is an arduous task to evaluate the dependability and prevent the failures. For

complex large scale systems, several errors may be triggered together and even several failures may occur simultaneously. It is necessary to evaluate the co-effect of these failures.

2.5.3 Technical challenges of run-time dependability management of SCPS

For SCPS, self-healing and self-adaptation are highly interrelated and mutually reinforcing. To simplify, the dependability of self-management can be divided into two parts: 1) the safety of decisions and the fitness of arranged candidates, 2) the reliability of infrastructures and the dependability of the procedure of decision process. Despite the traditional issues, the SCPS still faces six new challenges:

RQ1: How to obtain consistent observation sequences on all subsystems (in other words, how can all distributed subsystems reach a consensus on the timing behavior of events)? Temporal issues become extremely critical [34]. Reasoning needs the temporal information of events, different temporal orders of two events lead completely different conclusions of the (causal) relationships, wrong timing orders will mislead the DSS to make wrong decisions.

RQ2: How to make safe decisions with incomplete, inconsistent, inaccurate data and disordered events? In the real world system, data may be incomplete, messages and events may be delay and disorder. SCPS should still be able to make safe decisions and take right activities.

RQ3: How to continuously quantify the dependability of each subsystem under different contexts, especially in the unusual situation with rare evidences? The physical space changes continuously, while the cyber system generally is discrete. It is difficult for cyber system to seamlessly switch between two states. I.e. the reliability of subsystem s under the context c_1 at time t_1 is $p_r(s|c_1(t_1))$, and c_2 at time t_2 is $p_r(s|c_2(t_2))$, how to model the probability function during $t_1 \rightarrow t_2$ as we can't build a continuous model to describe $c_1 \rightarrow c_2$?

RQ4: How to keep consistent quality of service when a subsystem joins or quits? In other words, SCPS should guarantee the correctness and QoS of decision processing with different heterogeneous (but replaceable) subsystems under the unpredictable environment.

RQ5: How to tradeoff between dependability and other requirements dynamically? For SCPS, the distributed subsystems with limited resources should achieve real-time execution with minimum failure risk and energy consumption. Searching the optimal self-management solution is a typical dynamic optimization problem (DOP) [35].

RQ6: How to remove cumulative errors and continuously maintains the dependability of services, especially the self-dependability manager? As well-known as the Butterfly Effect, errors will accumulate with feedback loops. SCPS should remove the errors to keep the health of system. Making decision with unreliable evidence to remove errors suffers from the same problem of halting problem.

The RQ1 challenge is about the dependability of infrastructures (global reference time) and the reliability of data (events). RQ2 focuses on evaluation the safety risk of decisions. The challenges of RQ3, RQ4 and RQ5 are about the run-time dependability evaluation for self-adaptation. RQ6 is about the correctness and reliability of self-healing. These issues are intimately entwined with each other. Therefore to build a dependable SCPS, it needs systemic design and skillful compromise.

3. Dependable Self-adaptive architecture design

There is no doubt that architecture design plays an import role in SCPS engineering. In some senses, the upper bound capability of self-management depends on the architecture. When we chose one architecture, it always means that we have chosen a set of potential self-management strategies and given up others, i.e. distributed detection schemas are unsuitable for centralized architecture. Meanwhile, a good architecture can also reduce the complexity of SCPS, facilitate the arrangement of decision process and simplify the runtime maintenance, which can holistically improve the dependability of SCPS. In this section, we investigate the approaches on

architectures/frameworks of dependable SCPS in detail. Based on top-down analysis, we summarize the strengths and weaknesses of current SCPS architecture and propose a concept of hybrid architecture. Thereafter, we conclude about available technologies and methodologies to build a dependable SCPS. Finally, we summarize the shortage of current self-adaptive architecture. Here, we use the term "architecture" as the short of "architecture or framework".

3.1 State of the art of CPS architecture design and the key technologies

SCPS is an organic system with loose physical structure but compact logic organization. In the static view, all subsystems of SCPS are loosely decoupled and each subsystem could be regarded as an autonomous system. In the dynamic view, SCPS should make rigorous decisions and the subsystems should cooperate tightly to adapt to the intricate situations. To design such a SCPS, the architects should synthetically consider the intricate indexes, such as complexity, correctness, dependability, performance, scalability and flexibility and trade off among these requirements, which is a multi-objective optimization problem. Moreover, these indexes have nonlinear relationships with each other. Any fine-tuning may have deep effects on the abilities of SCPS.

As mentioned earlier, for SCPS, dependability issues are tightly intertwined with correctness issues. To build a dependable SCPS, we need to systematically consider the correctness and dependability (C&D). As CPS contains both discrete subsystems and continuous subsystems, to design the dependable architecture for CPS, we should pay special attention on the correctness of integration and the dependability of interaction between the discrete subsystems and the continuous subsystems. After analyzing 97 approaches, we selected 20 real (not conceptual) CPS architectures and summarized the key methods on C&D in those approaches in Table 1. These architectures can be classified into three types: Service Oriented Architecture (SOA) based architecture [36-38], Multi-Agent System (MAS) based architecture [13-14, 39], and other aspect oriented architectures like 5C architecture (5C stands for Connection, Conversion, Cyber, Cognition and Configure) [40], etc. In these architectures, the continuous subsystems are encapsulated into some special discrete subsystems, or special agents are developed to transform the signs and events between the two types of subsystems [40-41].

Generally speaking, SOA is mainly designed for the resource rich CPS with central manager system. It highlights the flexibility, composability and reusability of services, which can facilitate the dynamic service reorganization. However, *SOA is not suitable for all (subsystems of) CPS*. According to design principle, the services in SOA are **stateless** [42] and **location transparency** [42]. While *both the decisions and the actions in SCPS are state dependent and location aware*. Moreover, the interactions with physical space are safety critical, which need high observability and traceability for monitoring and controlling. Stateless services are unsuitable for composing these interactions. Furthermore, the SOA services have little autonomy and all of them are controlled by the central manager (the services choreographer), which implies that the system can't response to the accidents quickly. *Notice that the delay rapidly increases with the amount of involved services*. In summary, **SOA is more suitable for the stateless, delay-tolerant cyber services**, such as the long-term decision making services, and data services for the slow subsystems, i.e. human beings.

MAS is a popular architecture designed for larger scale distributed, decentralized SCPS. Every subsystem (agent) in MAS is an autonomous system, it can make its own decisions based on local context. But these decisions generally are very simple due to the limited resources. For MAS, swarm intelligence is the key technology to achieve self-management. However, high autonomy reduces the performance and controllability in somehow. The stochastic behaviors of autonomous subsystems also make it difficult to formally evaluate the effects of decisions, which reduce predictability and trustability. Moreover, the decentralized consensus solutions for MAS are still suffers the challenge of slow convergence rate [43], while decentralized consensus is the prerequisite to make dependable self-management decisions. Compared to SOA, MAS based decentralized solution needs much fewer resource and has higher robustness and faster response speed (though the response activities

may not be optimal). Generally speaking, *MAS is more suitable for the large scale geographically distributed system, especially whose subsystems have limited resources*, such as WSA.

Table 1. Current architecture researches on the dependable SCPS

Ref	Arch	low complexity self-adaptive and dependability ¹ means, etc
[36]	SOA based	Decouple, Compositional service; <i>Heartbeat, Real-time FDIR, Middleware based fault tolerate solution;</i>
[37]	SOA based	Unified Abstraction, Domain-Specific Description Schemas, Formal Semantics;
[38]	SOA based	Knowledge-Driven Service Orchestration, Ontology based service description;
[44]	SOA based	Formal contract for physical property, Dynamic physical behavior, Hybrid system behavior;
[45]	SOA based (ebbits)	Proxy, Virtualization, Middleware, Ontology, Semantic Knowledge, Rule base context recognition; <i>Predictive maintenance</i>
[14]	MAS based	Self-organizing & Self-adaptive models, Rules & Knowledge based Reasoning, proof-of-concept; <i>Exception Identification Model;</i>
[46]	MSA + Cloud	Data-driven self-organization, Intelligent negotiation based on contract net protocol, Deadlock prevention;
[41]	MSA + holons	Soft real-time MSA, Hard real-time function blocks (holons); <i>Redundancy;</i>
[47]	Cloud based	Virtualization, Multilevel smart scheduling algorithms; <i>Redundancy, checkpoints;</i>
[48]	Cloud based	Distribution middleware, Virtualized interrupt model, Spatial & temporal isolation based on partitioning; <i>Fault isolation;</i>
[49]	Cloud based	Virtualization, Task migration, Evolutionary algorithm for placement, WCET response time guarantee;
[50]	Software defined	Network-centered (SDN), Technology Standardization;
[40,51]	5C Arch	Decouple, knowledge based; <i>Prognostics and health management, Fault isolation & identification;</i>
[13,52,53]	Rainbow	Architecture-based self-adaptation (ABSA), (Re)scheduling, Strategy based, Mutation rules robustness tests;
[54,55,56,57]	DEECo	DSL, Decouple, IRM ² , Knowledge, Deterministic semantics, Formal analysis; <i>Proactive reasoning, Reliable communication;</i>
[58]	Na ³	Standardization, Open-Knowledge-Driven, Ontology;
[59]	Na	<i>8 steps comprehensive FDIR, Reliability Knowledge & Reasoning;</i>
[60]	Federation Arch	Component-based, Plug-in software, Plug-in run-time environment based on VM, Federation life-cycle management;
[61]	Na	<i>Fault mode, Reconfiguration, Rule based diagnosis, Reasoning;</i>
[62]	EVM ⁴	EVM, Virtual Component, EVM DSL ⁵ , Formal design, Multi-level & multi-object scheduling

¹The methods for dependability are shown in *italic*; ²IRM: Invariant Refinement Method; ³Na: no introduction is available; ⁴EVM: Embedded Virtual Machine. ⁵DSL: Domain Specific Language.

Moreover, considering the cost and risk of developing a new SCPS, some aspect oriented architectures are proposed to integrate the legacy systems into CPS. Generally, the developers assemble the computer systems or agents with legacy (industrial control) systems and connect them with industrial Ethernet. These aspect oriented architectures are generally dedicated to the special application (domain). Actually, the aspect oriented architectures are more a long term evolution solutions than the special architectures. Some proxy agents are designed to translate the protocols of legacy systems, and a new central manager system is integrated to coordinate these legacy systems. As the legacy systems are not well-designed for CPS, the central system can't collect detail enough information for decision making. As a result, it is also impossible for CPS to achieve fine grained control. Hence, *these architectures are not well fit for the large scale, complex SCPS*.

Strictly speaking, SOA and MAS are two kind of abstractions at different levels. The subsystems in MAS can be built with SOA if these subsystems have enough resources. I.e. the entire data center can be regarded as a powerful subsystem of MAS, and the data analysis services can be implemented with the SOA architecture. From this point of view, the hybrid architecture integrated with SOA and MAS is a promising solution for SCPS, especially if these subsystems provide the stateless services. For state dependent services, we still need to improve the services by patching the state information. Another graceful solution is redesigning the services and encapsulating them with some formal compositional abstraction, such as the actor. Actor is a generalized timed FSMs model, which supports cascade composition. For more detail introduction about actor, the reader is advised to refer to the book [63].

Besides flexibility and autonomy, controllability is another important requirement to the SCPS. An ideal SCPS (with hybrid self-adaptive architecture) should be able to grant the controllable autonomy to every subsystem. The autonomy should perfectly fit the role and ability of subsystems. The SCPS can adjust and revoke the autonomy when the situation changes. Technically speaking, **SCPS should decouple the architecture/topology control logic from the normal service logic**, and every subsystem should follow the same control specification. This is also the core design philosophy of SDN (Software Defined Network) [50] or programmable network [64], which tames the complexity through **decoupling** and **abstraction**. From this point of view, software defined architecture (SDA) or architecture-based self-adaptation (ABSA) [13,52,53] is a promising solution for SCPS. Both of them can significantly expand the potentialities of adaptation, which is useful for overcoming the challenges from **RQ1** to **RQ6**.

3.2 The methodologies to design a dependable SCPS

SCPS integrates multi-level technologies, and these technologies have various alternative solution, while the complexity of different integrations are quite different. To simplify the evaluation of the C&D of design, we need to select the most suitable technologies and organize them with right structure. Many exploratory studies have been done on this domain. Focusing on C&D, we analyzed the technologies/methods applied in the selected papers, which are shown in Table 1. These technologies/methods improve the C&D from diverse aspects.

Reducing the complexity (benefit for **MQ3**, **RQ1**, **RQ3**, and **RQ4**): **Decoupling** and **abstraction** are the two widely applied basic methods. Many successful cases have proved that their efficiency on taming the complexity and improving the flexibility and the maintainability. **Standardization** is another method to reduce the complexity of self-organization. It simplifies the orchestration of (heterogeneous) subsystems with concordant standards of behavior, such as the standard of interfaces and communication protocols. Decoupling and standardization are two native design philosophies, but can produce comprehensive benefits for architecture design, technology organization, and the strategies design. Though decoupling and standardization can't solve the problems directly, but they can systematically relieve the modeling problem **MQ3** and the dependability management problems **RQ1**, **RQ3** and **RQ4**.

Isolation and migration at task set level (for MQ2 and RQ4): **Virtualization** and **task migration** are two closely associated technologies, which are widely applied in cloud based CPS [45, 47, 49]. Thanks for the development of hardware, virtualization becomes an alternative solution for embedded systems [62]. There are two types of virtualization, one is virtual machine based virtualization, and another is container based virtualization. They respectively provide a virtualized, isolated runtime environment at subsystem level and application level. With virtualization solutions, the related tasks be organized as an isolated group and be migrated together between physical hosts. Through task migration, we can apply scheduling solution at task set level to balance the resource consumption among physical hosts and reduce the delay of data access. It enables the developers to design scheduling strategies without considering the detail constraints and dependency relationship among tasks. However, the prerequisite of task migration is that the tasks are location transparency, which limits the application scope to the data center (a.k.a. DSS). While to sensors and actuators, task migrated may be absurd. I.e. for the irrigation system, it is ridiculous to migrate the watering task from a dry place to a humid place. Compared with virtualization, **multi-role agent** is a finer-grained decoupling solution. Every subsystem of MAS can play one or several roles and even switch between different roles at runtime to meet the dynamical requirements of self-management, i.e. the actuator can play as a sensor to share its observation, or any subsystem can be a coordinator to coordinate the activities among neighbors.

Enhancing the dependability at architecture level (for RQ5 and RQ6): According to the analysis in Table 2, **temporal-spatial redundancy** is still the basic methodology to improve the reliability. Temporal redundancy technologies mainly include checkpoint and redo (log), which are mainly applied on DSS because of the time cost. To achieve spatial redundancy, virtualization is a new cost-effective solution to isolate the faults and prevent the error propagation. Task migration can improve the availability with warm standby or hot standby technologies. By using load balance technology, it can also reduce the risk of deadline missing and failure rates. Other spatial redundancy technologies include fault-tolerant protocols [65] and middleware [36], and multi-version, multi-copy (MVMC) [66], etc. Apart from the redundancy methods, we can also improve the dependability of SCPS with **design diversity** [66-69]. As an analogy with the importance of species diversity to the ecosystem, design is meaningful to SCPS. As it is impossible to test all code, the released system inevitably contains defects. It needs multi-version subsystems to tolerate the unexpected failures and improve the survivability of SCPS. Meanwhile, multi-objects decision making is NP-hard problem, diversiform optimization solution can increase the possibility to find the proper decision. However, some studies also show that diversity will increase the rate of undetected failures and design diversity takes positive effect only after certain quality threshold reached [68]. Considering the defect of redundancy based homogeneous solution (as seen in sub-section 2.5.1), we believe that *design diversity will an effective supplementary measure for building dependable SCPS*. One promising application of design diversity is to integrate different technical standards to compensate for each other's shortcomings. For example, Lidar, Radar and digital cameras are integrated together to enhance vision of self-driving car in different weather. Different wireless standards are integrated to balance the power consumption and transmission distance, such as IoT node integrates with IEEE 802.11 (Wi-Fi), IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee, 6LoWPAN), Sigfox, LoRa, WMBUS and NB-IoT [50, 70]. Heterogeneous CPU are integrated to balance the performance and the energy budget, i.e. subsystem may be integrated with FPGA, GPU, general CPU and low power MCU.

Improving the quality with formal model and formal analysis methods (for MQ1 to MQ5, and RQ2 to RQ4): To be self-manageable, SCPS should firstly model itself and analyze the statuses of both the cyber space and the physical space at run-time [71-73]. As shown in Table 2, we can design formal compositional architectures and formal quantifiable behavior models to improve the quality of SCPS, and to simplify the evaluation, especially the runtime self-evaluation, which can reduce the complexity of the model@run.time solution. It is a classic case of "using simplicity to control complexity". Meanwhile, formal/mathematical model@run.time solutions can simplify the

implementation, which can improve the correctness of analysis methods and reduce the side-effect of introducing model@run.time. Meanwhile, formal design overcome the drawbacks of design diversity, i.e. checking the consistency of behavior with isomorphic theory. And also we can apply compensation methods to keep the consistency of the nonfunctional requirements between different versions, i.e. using the reserved waiting time to align the execution time of two finite state machines.

3.3 Improve the dependability of self-adaptive architecture

Design a dependable self-adaptive architecture is the prerequisite to *overcoming the challenges RQ1 to RQ6*. A good self-adaptive architecture can significantly widen the design space of self-management strategies, simplify the orchestration of services, facilitate the interactions between subsystems, clarify the causality of events, reduce the complexity of system evaluation, and boost the performance of self-management.

3.3.1 Simplify self-management with hybrid self-adaptive architecture

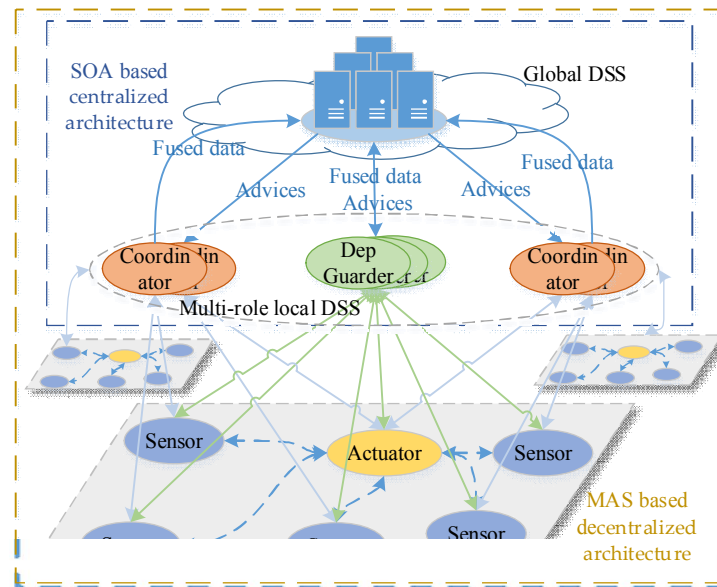


Figure 5. The concept of hybrid multi-role self-adaptive architecture

The popular solutions of self-adaptation architecture include architecture-based approaches, multi-agent based approaches and self-organizing based approaches [74]. Due to limitation of current technologies, single self-adaptation architecture can't well meet the requirements of large scale (geographically) distribution, smart and quick reactive. Obviously, most of subsystems, i.e. sensors and actuators, are not powerful enough to process data driven decision making (i.e. Machine learning based technologies). And it is also impossible to deploy all services on one single subsystem. Hence, we need to define the role for subsystems and divide their works, to improve the performance and reduce the global complexity (the detailed analysis is seen in subsection the subsection 3.3.4). Meanwhile, it takes too much time to transmit data from sensors and actuators to the data center, which reduces the timeliness of decision making. The lagging decisions may not be able to catch up with the changes of the real world. Hence, neither decentralized architecture (i.e. Multi-agent based approaches) nor centralized architecture (i.e. SOA based approaches) can well meet the requirements of intelligence and response speed. Hence, hybrid architecture with multi-term self-management is the only reasonable compromise, which is shown in Figure 5. To achieve both high intelligence and real-time reaction, we also need an efficient multi-level and multi-term feedback loop to achieve real-time self-adaptation, which is shown in Figure 6 (to simplify, only two levels of loops are shown in this figure).

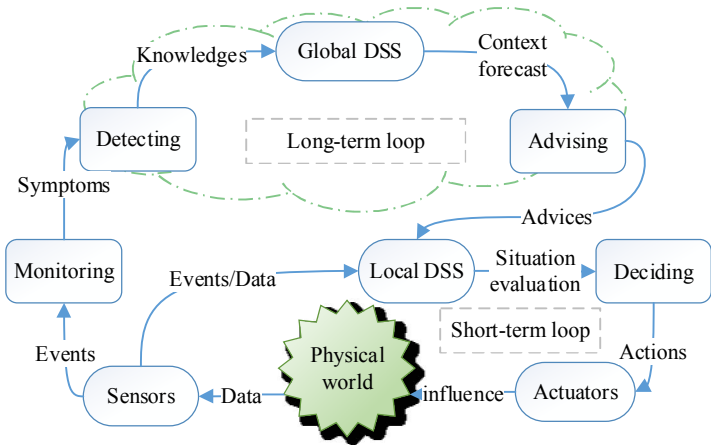


Figure 6. Multi-level and multi-term feedback loop[73].

With hybrid multi-level architecture, the global DSS can use the big data analysis technologies and “AI” technologies to make wiser, more farsighted self-adaptation decision. The powerful cloud system also make it possible to systematically analyze the safety of prophetic adaptation decision with multiple complex evaluation methods from different angles (as seen in the subsection 5.2), then generate dependable advices to process the proactive decision. The local DSS can refine the constraints of advices with newest observations, select the proper candidates of decision executors and organize them in right order to process in the optimal way. For some emergencies, the actuators can take actions based the rules without waiting the decision from the DSS. With multi-level and multi-term feedback loop design, the SCPS can achieve high intelligence as well as real-time response. Meanwhile, the hybrid multi-level architecture can also tolerate the failure of partial subsystems, which improves the dependability of SCPS [75], Through elaborated integration of decentralized and centralized architectures at different levels, the hybrid multi-level architecture can overcome the shortages of them and achieve high performance, scalability and flexibility. Meanwhile, hybrid architecture is also a long term evolution solution, which can well integrate the legacy systems.

3.3.2 Guarantee the timing dependability of events for reasoning (for RQ1)

Guaranteeing the consistency of timing behavior is a systemic challenge for (large scale) distributed parallel systems [34, 76]. To SCPS engineering, the timing dependability is more serious and urgent [34, 40, 77], because the applied reasoning methods are based on the analysis results of the causal relationships of events. And timing order is one key index of the causal relationships. If SCPS can't reproduce the timing relationships of events correctly, it will generate wrong view of the statues of cyber space and physical space, and get wrong analytical results, which will mislead the DSS and self-healing manager, even lead to disasters in some cases.

The precision of timing depends on massive of factors, such as the quality of hardware (clock), the dependability of software, and the software control strategies. Due to the limitation of current technologies, i.e. the deviation between crystal oscillators, the stochastic jitter of signals, etc. [34, 76], it is very difficult to achieve the precise global reference time (absolute time) in distributed CPS. What's worse, timing has never been treated as a basic premise ago [77], which implies that the legacy hardware and software systems can't well support the requirement of precise timing. Without dependable information of timing behavior, it is impossible to generate dependable self-management decisions, let alone dependable SCPS. Fortunately, more and more researchers aware the importance of timing and try to redesign the whole system from hardware platforms [78] to operating system [79], from programming models [80] to architectures [81], and from time synchronization protocols [82] to communication standards [83-85]. These studies mainly focus on improving the precision of the timestamp of events, the timeliness of actions (a.k.a. taking actions in

right time) and the stability of the time of communications and decision processes (i.e. decrease the difference between worst case execution time and best case execution time).

Timing dependability is an open issue to SCPS. With the increasing scale and autonomy, it becomes more and more difficult to reach a consensus about timing behavior. One natural idea is to build a new programming model with embedded temporal semantics [73, 86]. The conceptual programming model is illustrated in Figure 7. For each task or decision, SCPS will generate a decomposable contract with timing and dependability requirements. Every subsystem should strictly follow the contract and process the related activities on time and in time. To tolerate unexpected delay, it can reserve some time for each activity. Meanwhile, several redundant subsystems are arranged together to process the decisions together to minimize the risk and improve timing dependability.

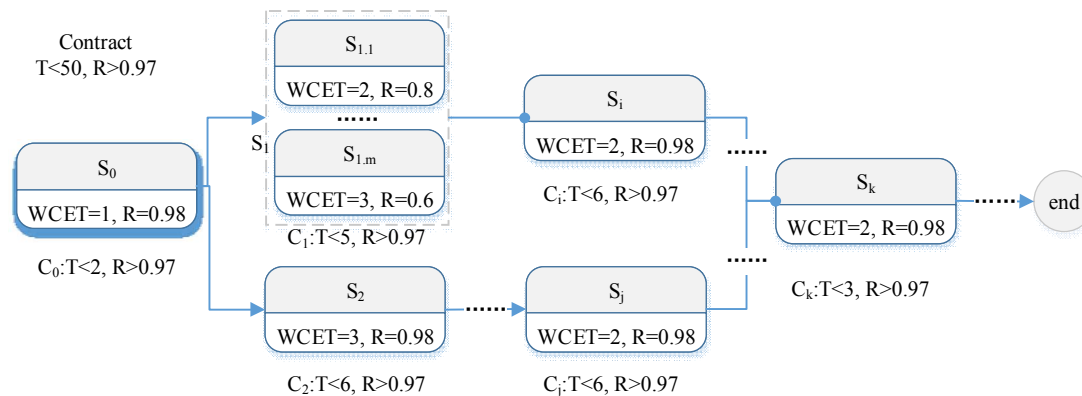


Figure 7. The concept model to guarantee the timing dependability with decomposable contract

Take the case in Figure 7 as an example, there is a decision with the contract whose maximal process time is 50, and the reliability is greater than 0.97 ($T < 50$, and $R > 0.97$). For the subsystem S_2 , its worst case execution time (WCET) is 3 and its reliability is 0.98, the decomposed contract for S_2 is ($T < 6$ and $R > 0.97$), the reserved time is $6 - 3 = 3$. For the subsystem set S_1 , as no single subsystem can satisfy the contract ($0.8 < 0.97$ and $0.6 < 0.97$), m redundant subsystems are arranged to process the decision together. The results of redundant operations and parallel operations should be synchronized when they are finished (the synchronization is shown with dotted arrows, such as the input arrows of S_k in Figure 7).

Though the contract based programming model can achieve high flexibility as well as controllability and predictability, it is not compatible with legacy systems. To apply this model, we should redesign all subsystems, also get the precise values of the best case execution time (BCET) and WCET, and narrow the range of [BCET, WCET]. Meanwhile, the model also needs precise real execution time. To satisfy this constraint, we should design precision timed infrastructures, which is still an open issue [76]. To get precise BCET and WCET, we also need platform dedicated tools (which will be introduced in section 4), and the related methods to alleviate the time analysis problems that discussed in [87–88]. Moreover, we need effective run-time reliability evaluation methods to avoid misleading estimates, and to avoid applying wrong fault-tolerance scheduling strategies [89].

3.3.3 Improving composability and compositionality of services (for RQ1 to RQ6 and MQ1 to MQ5)

To well adapt to the changeable environment, SCPS should select the most proper subsystems, organize them with the right topology, and arrange them to process the decision in right order. To cooperate smoothly, subsystems should not only share the interoperable interfaces and identical data specification but also provide service in a consistent way, i.e. if subsystem A is replaced by

subsystem B for some reasons, B should not only continue the services, but also provide them with the same quality as A does, or at least not worse than A. Only in this way can SCPS provide seamless services reorganization and process the decisions dependably. In other words, SCPS should evaluate the differences of properties between subsystem A and B, and guarantee that the replacement will not introduce any intolerable inconsistency (i.e. executing different if-else branches, because the conditions of decision may be near the critical value). To tame the complexity of the arrangement and related evaluation, we should improve the *composability* and *compositionality* (C&C) of subsystems and decisions [5, 90-91]. So that a complex decision can be decomposed into several small subtasks and processed independently; as well, the global behavior can be kept consistent with the original solution. Meanwhile, the subsystems with good C&C can cooperate and switch smoothly, which is meaningful to maintenance.

Composability is the quality that component properties do not change by virtue of interactions with other components [91]. It originates from the philosophy of reductionism, which highlights the consistent behavior of the subsystem when it cooperates with other subsystems to build a larger system. On the contrary, **compositionality** is originated from holism. It is the ability that system level properties can be computed from and decomposed into component properties [91]. Compositionality is more about the capacity to decompose the system level properties. It focuses on the consistency between the system level properties and the divided properties (component properties), where the system level properties can be reasoned with the properties of components/subsystems. More detailed discussion about C&C refers to [91]. By the way, the concepts of C&C are interchangeable in some researches.

Improving C&C can symmetrically and significantly promote the quality of SCPS, which attracts more and more attentions. As heterogeneity of abstraction layers leads to loss of predictability of system behavior, through adding additional semantic information of interactions, Sztipanovits et al. presented a passivity-based approach to decouple system stability from cyber timing uncertainties [92]. Focusing on the vertical composition, the approach explored the systematic development method for CPS integration [92]. Nuzzo P. et al. adopt contracts based components specification and abstraction, and provided a platform-based design methodology with formal supporting for the entire CPS design flow [93]. Paul A. et al. proposed a general formal framework for architecture composability based on an associative, commutative and idempotent architecture composition operator; the authors established preservation of safety and liveness properties by adopting architecture composition [94]. Aiming at constructing a compositional proof of correctness, Sanjit A.S presented a formal methodology and a theoretical verification and synthesis framework integrating inductive learning with deductive reasoning [95]. Stavros T. detailed the key principles of compositionality focusing on interface design for MDD [96]. A compositional specification theory for components reasoning is proposed in [97], this specification theory uses synchronistic input and output (I/O) actions to abstract the temporal ordering of behavior. To ensure the interoperability of components and adherence to specifications, a generic algebraic method is developed and two strategies are proposed to synthesize or refine a set of contracts to evaluate the composition satisfaction of a given contract [98]. Moreover, the authors also presented a set of decomposing conditions for verifying the decomposition of a contract into a set of contracts [98].

Currently, C&C are mainly researched as a part of model driven engineering/development (MDE, MDD). Considering the similarities of the model at design period and at run-time, we proposed a formal decentralized compositional framework and developed a compositional actor base prototype system to explore the value of C&C to SCPS [73]. As SCPS is a kind of SoS, we believe that improving C&C of subsystems can comprehensively reduce the evaluation complexity of run-time composition and guarantee the C&D of self-management. Currently, two types of solutions are proposed to guarantee quality of composited service at runtime: **one lightweight solution to focus on the invariant requirements** [99], **another universal solution** (as shown in Figure 7) **is to build the formal mapping and calculation rules between the requirements/contracts and the**

properties of subsystems [73]. A SCPS with good C&C can comprehensively, systematically simplify the (model@run.time based) evaluation of runtime composition and arrangement of candidates, and improve the quality of self-management. Overall, *improving the C&C of SCPS is a key solution to overcome the challenges RQ1 to RQ6 and MQ1 to MQ5.*

3.3.4 Improve the dependability with SDA (for MQ3 and reducing complexity)

As we all know, complexity negatively affects the dependability of system [29]. However, with the increasing scale of SCPS, we have to deal with the rapidly increasing complexity. As mentioned above, software defined solution, which is named as SDN or SDA or ABSA, is one promising approach to tame the complexity. In this subsection, we formally explain how to reduce the complexity and improve the dependability of system with SDA (a.k.a. decoupling the (architecture) control logic from normal functional logic).

To SCPS, we have 3 types of available organization, which are shown in Figure 8. Let us suppose that there are n subsystems $\{A_1, \dots, A_n\}$. In the schema (c), there is one additional subsystem A_m to manage the structure/topology control logic. To simplify, we assume that the reliability of all subsystems are the same, and just take the reliability of software into account. We adopt the commonly used exponential reliability function $R(t) = e^{-\lambda t}$, where λ is the failure rate. To simplify, we assume λ is proportional to the software complexity C , $\lambda = kC$ where k is a scaling constant (k depends on the development effort) [29]; that is $R(t) = e^{-kCt}$. Let us normalize the mission duration t to 1 and let the scaling constant $k=1$. As a result, we can rewrite the reliability function with a normalized mission duration in the form $R(c) = e^{-C}$.

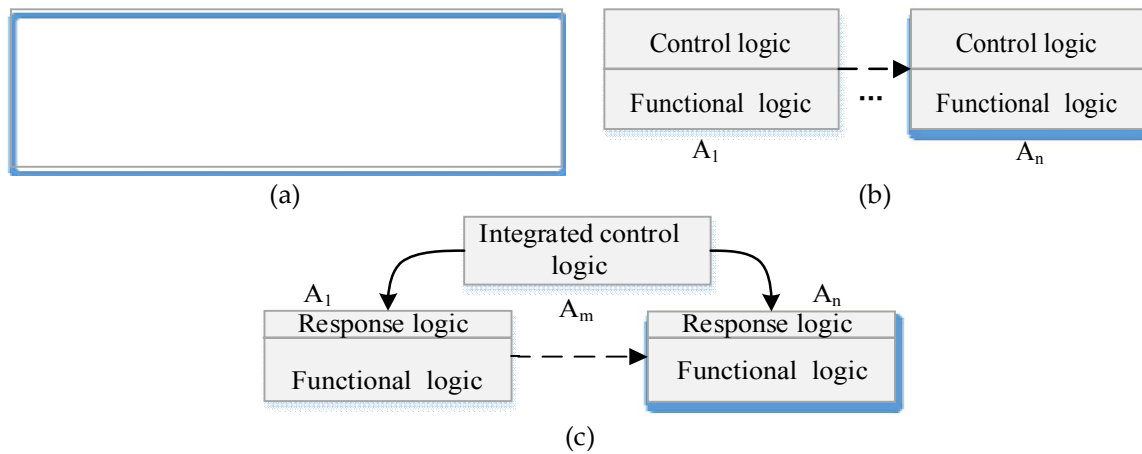


Figure 8. The schemas of decoupling: a) monolithic design, b) logic level decoupling, c) role level decoupling with physical isolation. (*the arrow represents the direction of control logic flow)

Let us denote C_i^f as the complexity of function logic (business logic) of A_i and C_i^c as the complexity of control logic of self-organization. The complexity of A_i in schema (a) is C_i^{f+c} (business logic and control logic are not decoupled). Let us assume that the decoupling effects (decoupling and abstraction) are correct and effective. Hence, we can reduce the complexity of system. Formally, we have $C_i^{f+c} > C_i^f + C_i^c$, (actually, software has at least polynomial complexity, i.e. according to the Agresti-Card-Glass system complexity metric [100], the global complexity of a software is $G_t = S_t + D_t$, where $S_t = O(\sum ((f_i)^2))$ is the simplified Henry-Kafura information flow complexity metrics for modular without fan-in, while the original metrics is $S_t = \text{length}_t * (f_i(\text{in}) * f_i(\text{out}))^2$ [101], $f_i(\text{in})$ and $f_i(\text{out})$ are the fan-in and fan-out of the module A_i ; $D_t = (\sum V_i / (f_i + 1)) / n$) is the mean of all the internal complexity measures for all of the modules,

which is similar to the Halstead's metrics [102], V_i is a function of the amount of internal variables). In the schema (c), the control logics are moved to the manager A_m , A_i keeps fewer logic to response to the commands from A_m . Notice that, A_i can reuse the communication code. Let's assume that the additional complexity of response logic of A_i is C_i^r . For a dynamic system like SCPS, we have $C_i^r < C_i^c$. The complexity of A_m is C_m^c . Let us assume that $C_m^c = \sum_{i=1}^n (\beta_i * C_i^c)$, where β_i is a scale factor that depends on the reuse rate of control logic of A_i , $\beta_i \in (0,1)$.

The reliability of A_i in the schema (a) is $R(A_i) = e^{-C_i^{f+c}}$, and the global reliability is $R(a) = \prod_{i=1}^n e^{-C_i^{f+c}} = e^{-\sum_{i=1}^n C_i^{f+c}}$. For A_i in the schema (b), its reliability is $R(A_i) = e^{-C_i^f} * e^{-C_i^c} = e^{-(C_i^f + C_i^c)}$, and the global reliability is $R(b) = \prod_{i=1}^n (e^{-C_i^f} * e^{-C_i^c}) = e^{-\sum_{i=1}^n (C_i^f + C_i^c)}$. The reliability of schema (c) is $R(c) = e^{-C_m^c} * \prod_{i=1}^n e^{-C_i^f - C_i^r} = e^{-(C_m^c + \sum_{i=1}^n (C_i^f + C_i^r))}$.

Improving reliability: 1) As $C_i^{f+c} > C_i^f + C_i^c$, obviously, we have $R(b) > R(a)$. 2) We have $R(c) > R(b)$ if we can achieve $\beta_i \leq 1 - C_i^r / C_i^c$ through decoupling. Proof. the inequality can be transformed into $\beta_i * C_i^c + C_i^r \leq C_i^c$, then we have $\sum_{i=1}^n (\beta_i * C_i^c + C_i^r) + \sum_{i=1}^n C_i^f \leq \sum_{i=1}^n C_i^c + \sum_{i=1}^n C_i^f$. With $C_m^c = \sum_{i=1}^n (\beta_i * C_i^c)$, we have $C_m^c + \sum_{i=1}^n (C_i^r + C_i^f) \leq \sum_{i=1}^n (C_i^c + C_i^f)$. Because $R = e^{-C}$ is a strictly monotone decreasing function, we have $R(c) > R(b)$. As self-healing is also a type of control logic, external self-healing is recommended for complex SCPS (conclusion for subsection 2.4). Notice that, $\beta_i \leq 1 - C_i^r / C_i^c$ is a strong condition. In practice, it is acceptable that some subsystems don't meet the condition.

From the view of structure/topology reorganization, most subsystems of SCPS, especially these redundant subsystems, have similar control logics/operations, i.e. finding the best successor, disconnecting, waiting and synchronizing. With the increasing of the scale, SCPS has more similar subsystems and it is easier to have small β_i . The reliability equations show that it can improve the reliability of SCPS by decoupling the similar control logics and moving them to a special manager subsystem. SDA can simplify the management of version consistency and improve the utilization of resource. Moreover, SDA can also reduce the differences in behavior that caused by diversity, which can improve the stability and predictability. The greatest advantage of SAD is that we can design a formal programmable contract, and the manager A_m can adjust the behavior of subsystems according to the contract. Furthermore, the coordinator subsystem can generate a new service to coordinate the temporary team. As most of contract validation are the same, i.e. the grammar checking, and the compilation (if we use bytecode supported language like the java), the integration of these codes can significantly improve the dependability of contract and simplify the normal functional logic.

3.4 Summary of the dependable SCPS architecture and organization

In this section, we surveyed the technologies/methodologies to design a dependable SCPS at the architecture level and organization level. Obviously, it is impossible to achieve the dynamic adaptation in changeable environment with static architecture and predefined rules. Hence, we need to design dynamic and flexible architecture for SCPS. With dynamic architecture, SCPS can generate the optimal solution for each adaptation decision by selecting the best subsystems and arranging them with an optimal organization. For engineering, we need to follow the methodologies of decoupling, abstraction and design diversity, design a formal compositional architecture, and improve the C&C of heterogeneous subsystems. Therefore, we can be able to build a dependable SCPS.

Reducing complexity and using simplicity to control complexity are the core methodology to improve the dependable SCPS at architecture level. *Decoupling* and *abstraction* are the cornerstones of these technologies and are widely applied in architecture design. With the increasing scale of SCPS, organization management becomes more and more important. *Decoupling the control logic* of self-organization from functional/business logic is an inevitable trend, such as ABSA and SDA. Considering the limited resources of a single subsystem, the roles of subsystem will be subdivided and subsystems just focus on partial tasks, such as fault detection and fault diagnose. The architecture with *multi-role* subsystems will be a natural solution to SCPS. Hence, we can design heterogeneous subsystems with specified infrastructures (hardware and software) to perfectly match the requirements of roles.

Redundancy and fault isolation are the two basic solutions to improve the dependability of system. For the subsystems with rich resources (i.e. DSS), virtualization is a cheap and efficient solution to implement them. For the sensors and actuators, the common solution to achieve fault tolerance is deploying redundant physical devices. Another popular method is design diversity, which is adopted to tolerate the failures that caused by the unexpected environment and defect of software. Task migration/scheduling is the basic technology for self-healing (i.e. Fault recovery and fault avoidance). Currently, Rules, knowledge and ontology are three mainstream methods to implement self-management decision making. Data driven methods, especially machine learning based methods, are becoming more and more popular for self-adaptation. However, the researches of data driven self-healing are few. One main reason is that current data analysis theory is not appropriate for analyzing the causality, and more detailed discussion will be found in section 6.

Redundancy with the same subsystems can't avoid the failures caused by the defects of design. To tolerate the unexpected failures and improve the survivability of SCPS, we need to apply diversiform design. We believe that *design diversity is the inevitable trends* for SCPS engineering. However, design diversity increases the inconsistency of the heterogeneous subsystems. It is recommended to *apply design diversity with standardization and formal abstraction*. Standardization can provide consistent definition of interfaces and message format, and formal abstraction can generate diversiform but isomorphic subsystems, which are two useful solution to avoid ambiguous behaviors. *Isolation* is another remedial measure to reduce the effect of ambiguous behaviors.

To improve quality of services, the nonfunctional requirements should be quantifiable and decomposable, so that subsystems can clarify their roles and responsibilities. The alternative solutions are *improving the C&C* of nonfunctional requirements and *building quantitative evaluation functions* between requirements and the properties of subsystems. Only in this way, can SCPS be able to decompose the decisions and arrange the best suitable subsystems to process them; and can the subsystems perform their own duty and meet the global requirements. However, the C&C of legacy systems generally are not very good. To build a dependable SCPS with legacy systems, we need to redesign or repackage the legacy systems to improve their C&C..

We believe that good C&C is a necessary feature for SCPS and all subsystems. Because a subsystem with good C&C can smoothly interact with other subsystems. A decision with good C&C can be easily decomposed to sub-decisions and processed accurately by subsystems. We also believe that formal architecture is a good solution to improve the C&C of SCPS. It can simplify dependability V&V of design and the risk evaluation of self-management decisions. Moreover, formal architecture is also an efficient solution to achieve high C&C. It can simplify the V&V of design and the evaluation of decision, and fault detection and diagnosis. What's more, formalization can also improve the controllability and predictability of the behavior of SCSP.

4. Guarantee the dependability of the design with model based V&V

There is no doubt that models play important roles in both SCPS design and runtime decision evaluation. Model based V&V is a necessary measure to guarantee quality of SCPS design. With model based dependability V&V, we can point out the vulnerable subsystems and remove the

defects as earlier as possible. As dependability solutions will introduce additional complexity, we should evaluate the efficiency and side effects of these solutions, and then compromise among these objects. Meanwhile, the models and V&V methods can also be applied for model@run.time evaluation to guarantee the C&D of self-management decisions. In this section, we review the current researches on model based dependability V&V of SCPS (dynamic system). Thereafter, we analyze the challenges of the current modeling tools and propose a concept solution of MDE based dependability V&V.

4.1 Current researches on model based dependability V&V for SCPS

SCPS is safety critical, we cannot emphasize the importance of dependability of SCPS too much. There are massive of dependability V&V models/methods, such as Fault Tree Analysis (FTA), Binary Decision Diagram (BDD), Reliability Graphs, Failure Modes and Effects Analysis (FMEA), Failure Modes Effects and Critical Analysis (FMECA), Hazard and Operability Analysis (HAZOP), Functional Hazard Assessment (FHA), Risk Reduction Analysis (RRA), Markov chain analysis, Petri Net analysis etc. However, the traditional methods are designed for the system with static architecture, which are not powerful enough for evaluating the dependability of SCPS. In this survey, we focus on the dependability evaluation methods for dynamic system (SCPS can be considered as a kind of dynamic system), and the detailed introduction of traditional dependability V&V methods refers to [103-110].

To analyze the random failures of the dynamic behavior, simulation model seems the only effective V&V method. Generally speaking, there are three types of solution to build a simulation model. One is building the model with general purpose programming language, i.e. the network simulators (NS3), which is the most powerful solution. However, *with the increasing complexity, it becomes more and more difficult to evaluate the correctness of the model itself*. The second solution is extending the traditional dependability models by appending dynamic temporal semantics and structure description semantics. For example, A formal modeling approach with object-oriented Petri nets is proposed for High-Confidence CPS [111]. However, *the extended models are still not powerful enough to analyze the detailed adaptation behaviors*. These solutions can't model the behavior that one subsystem joints the leaves the makeshift team, and it is also difficult to model the correlation distribution between inputs and outputs.

The third solution becomes more and more popular, it tries to build dedicated dependability models with more powerful formal languages, and the languages should support to model the dynamic behavior, i.e. high level dynamic formal language [112] (such as DEPICT [113] and Architecture Analysis & Design Language (AADL)[114]). To quantify the impact of unavailability of SCADA (Supervisory Control And Data Acquisition) system, a mathematical method combined with mean failure cost (MFC) metric and the classic availability formulation is proposed [115]. A new set of solutions based on undirected graph is proposed to verify the topology verification and locate the failures in Software Defined Networks (SDN) [116]. A dedicated symbolic model based on a combination of existing notions of robustness is introduced for designing robust CPS [117]. To overcome the incomparable of the dual conjunctive scheme, a method is developed for verifying N-inference diagnosability [118]. To evaluate the ability of withstanding and recovering from irresistible catastrophic, Alexander A. G introduced the concept of resilience and present two classes of formal models, which are multi-level directed acyclic graphs and interdependent coupled networks [119]. To formalize and assess the reconfigurable CPS, Linas L. et al. formally defined a dynamic system architecture in Event-B, and verified the correctness based on derived model. To guarantee the resilience of data processing, the authors transformed the Event-B model to the statistical UPPAAL and evaluated the likelihood of resilience under different system parameters with a statistical model checking [120]. To analyze the unpredictable and random behavior, the higher-order logic (HOL) is employed to model cyber-physical transportation systems formally. The authors obtained a randomized model by adding appropriate random variables and the probability

or expectation of the system based on formal reasoning [121]. Taking TTA, FlexRay, and TTCAN as examples, I. Saha et al. analyzed the three startup algorithms of time-triggered architectures and presented specified formal models of faults for these architectures, then proposed different verification approaches for the three startup algorithms [65]. Compared with other two methods, formal behavior descriptive languages achieve a good balance between fidelity and performance. Consequently, we can build the model in almost the same detail with the general purpose programming language, and guarantee the correctness of the model by using formal evaluation.

4.2 Improving the trustability of the dependability V&V results with cross validation

Table 2. Recent researches on MDE/model based SCPS dependability V&V

Ref	Type ¹	V&V ²	Key analysis technologies
[122]	RCA	R	Markov Chain (MC)
[123]	RCA	R	RBD, MC, Monte-Carlo simulation
[124]	RCA	R, A, ST	Stochastic Petri Net
[125]	RCA	D	MC, Stochastic Activity Network
[126]	M2M	D	Dependability domain ontology, FMEA
[127-128]	M2M	C, D	NuSMV, FTA, FMEA, HSIA, MC
[129]	M2M	C, A, Rb	BDD, BMC, FTA, FMEA
[130]	M2M	C, SF	Probabilistic temporal logic language, MC
[131]	M2M	R, A, M, SF	Bayesian Belief Network
[56]	M2S	SF, Rb	Simulation and statistical analysis
[132]	M2S	C, R	Automata-based diagnosis, LTL (Linear time Temporal Logic) based contract checking
[133]	M2S	C, Rb, R, SC	Calculation with mathematical model & Simulation
[134]	M2S	SF	Simulation and statistical analysis

¹ RCA(root-cause analysis) means analyzing the dependability with original dependable model (without transforming); M2M (model to model transformation), which is the technology to transform the reference model to the RCA models; Sim is short for analyzing with original simulation model; M2S (model to simulation transformation), which means generating the simulation model from reference model; ² Available items: Correctness (C), Dependability (D), Reliability(R), Availability(A), Safety (SF), Sustainability (ST), Robustness (Rb), Maintainability (M), Security (SC)

Currently, no language is powerful enough to model and analyze the behavior of SCPS at all levels. Moreover, with the increasing complexity of the models, it becomes more and more difficult to evaluate the correctness and the fidelity of models. To overcome these issues, we can apply **cross validation is necessary to check the correctness of simulation results**. Hence, integrated solutions become more and more popular. After reviewing 221 papers on model based dependability V&V, we selected 14 papers (belongs to 13 approaches) with relatively detail introduction to their methods, which are shown in Table 2. There are three types of model integrated solutions, which are root-cause analysis (RAC), model to (formal) model (M2M) and model to simulation (M2S). For the RAC solution, several (traditional) formal models or dedicated models are applied independently to analyze the possibility of failures and the root causes behind the failure. Generally, the models of RAC work at different levels. M2M is an advanced RAC method. It just needs to build a meta-model, and then the toolset will automatically transform the meta-model to the RAC model. M2S solution also needs to build a meta-model then the meta-model will be generated into simulation models. Both M2M solution and M2S solution can improve the efficiency of modeling, as well as the consistency of models, which are meaningful for cross validation.

Dependability is an integrating property. As shown in Table 2, MDE based solution can applied to evaluate the SCPS from various aspects, such as correctness, reliability and availability. To build a dependable SCPS, we should go a step further, comprehensively analyze the upper bound and the low bound of these indexes with considering the reorganization of architecture, and draw the scope of capability of different organizations, and build a straightforward mapping between situations and its most suitable organization. However, to our best of knowledge, there is no model or toolset, which is powerful enough to comprehensively model and analyze the whole SCPS. Currently, many conference papers are published to introduce the explorations on toolset development [135-139]. However, these works are at the beginning, it takes time to integrate tools and validate the correctness of model transformation. As how to develop powerful MDE toolset is beyond the scope of this survey, we don't discuss them in detail.

4.3 The challenges of model based dependability V&V

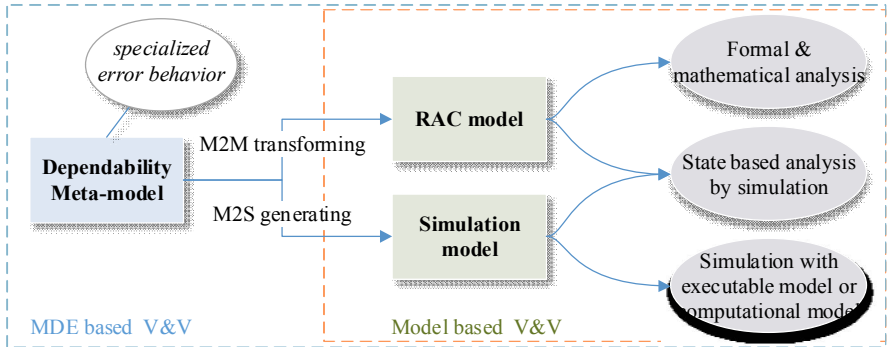


Figure 9. The general model transformation flow of model based dependability V&V

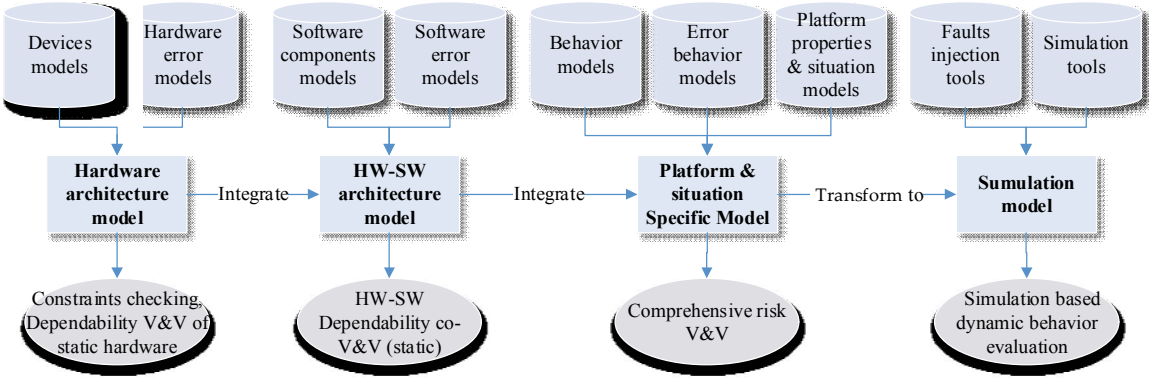


Figure 10. The concept solution of incremental dependability V&V for SCPS engineering: An AADL based MDE view

The general model transformation flow of the integrated solutions is shown in Figure 9. For an ideal integrated solution, designers just need to build the meta-model and set related platform dependent properties, and then the toolchains can automatically finish the remaining work, such as generating the analytical models and simulation models, and generating the related dependability and weakness analysis repots. The MDE flow with an ideal toolchain is shown in Figure 10. To build a useful toolchain for SCPS dependability V&V, in addition to the tremendous development efforts, we still need to overcome four challenges. 1). *Specialize the error behavior model*: developers should build a library of errors, sign all original sources of errors and define the trigger conditions and the occurrence probability of each faults; and for the safety-critical system evaluation, the consequence of failures should be given. 2). *Embedded the specialized error behavior model into computation integrated model*: developers should bind the states and variables between normal and

error models, define the connection of normal behavior and error behavior. 3). *Transform the dependable system model to RCA models or simulation model*, i.e. transforming the hybrid AADL behavior model to petri net model. 4). *Developing compositional libraries and supporting incremental validation*. SCPS contains massive (heterogeneous) subsystem. The costs is too high to evaluate all possible combinations. The compositional libraries are necessary and the models should support incremental validation. Only in this way can we efficiently verify SCPS with MDE solutions.

MDE is an ongoing research to improve the quality of design and the efficiency of SCPS development. The main advantages of MDE toolset is that it can automatically transform one meta-model to one or several detail models, and then analyze the dependability with the current formal methods or simulation tools. Through formal model transformation, MDE can guarantee the consistency between meta-model and other analytical models, and improves the trustability of V&V results with cross verification. Model based on SCPS (dependability) V&V becomes more and more popular [140-142]. MDE can help the designers to find defects at design period and reduce the cost of development on the domain of IoTPS (Internet of Things People and Services) [45, 143]. The exploration researches have shown remarkable advantages in developing complex software for industrial applications [144]. However, MDE based SCPS V&V is still at the beginning stage. Up to data 2018-8-30, only 4 MDE toolchains have been proposed to improve the dependability of SCPS. One is about statistical model checking with considering stochastic delays and errors [145] and another 3 proposals focus on MDE based debugging and testing [146-148]. As MDE is a vast independent field, the detail discussion of MDE refers to the comprehensive survey [149]. The authors investigates 10 MDE approaches on high reliable/available component-based information and communication system from 13 characteristics. Another detail survey on model based root-cause analysis (RCA) refers to [150], and more basic introduction of MDE on embedded system development refer to [151-154].

4.4 Brief summary and discussion

Models, especially formal dynamic models, play so important roles in CPS V&V and maintenance, that Lee even argued that "CPS theory is all about models"[155]. However, most of current V&V tools are closed source and have different input and output formats, even incompatible parameter sets. It is lack of powerful MDE toolset to transform the meta-model to these V&V tools. Moreover, the model theory for dynamic system is just at the beginning, powerful V&V tools are necessary to reduce the complexity of V&V. General speaking, probabilistic model is the mainstream for modelling the dependability of the dynamic system, i.e. Markov chain model [123], Bayesian networks [156]. However, it is infeasible to enumerate all available possibility for a complex dependability model. The analysis results are doubtful if we don't have the accurate probability of failures. Unfortunately, it is hard to obtain the accurate failure distribution of failures (MQ1).

With the increasing complexity of models and tools, it becomes a big challenge of how to guarantee the correctness of models and ensure the fidelity of simulation. Current modeling theories and simulation tools are unable to well handle the complexity of SCPS. To effectively evaluate the dependability of dynamic behavior (self-management), we should improve both modeling theory and simulation toolchains. Traditionally, we can describe the subsystem of SCPS with four type of models, which are discrete model (i.e. event-driven model), continuous model (i.e. control model), probabilistic model (i.e. the models analyzing with Monte Carlo simulation, such as Bayesian networks model) and deterministic model (i.e. physical model and deterministic automata). These models describe the system from different perspectives, but no single type of model can cover all perspectives. It seems impossible to comprehensively model the dynamic behavior of SCPS with single technology. While current integrated V&V tools can't well manage the interactions between different models. There are two kinds of directions to overcome these issues. one is exploring the organically integration of multi-types of models (i.e. the exploration of co-simulation

[157]). Another solution is building a meta-model, and transforming it into other proper analytical models (i.e. AADL based toolset [127]), and then improving the trustability of results with cross validation. To guarantee the correctness of models and reliability of simulation, we need to guarantee the consistency of model communication for the first approach, and guarantee the consistency of model transformation for the second solution.

Moreover, with the increasing amount of subsystems, the complexity of model analysis grows exponentially (i.e. State space explosion, which is a NP hard problem). Improving C&C of subsystems' model is the only available solution for this issue. An ideal MDE tool should not only be able to decompose the SCPS model into several sub-models and analyze them separately, but also evaluate the fault propagation between sub-models and compose the results without reanalysis the sub-models. To support composition, the MDE tool should be able to evaluate the comprehensive effects of multi-error sources. With such a MDE tool, we can analyze the dependability and other qualities of the whole SCPS in a formal and compositional way.

5. The safety of self-adaptation strategies and the dependability of real-time decision process

As SCPS can permanently change the physical world, we cannot emphasize too much the dependability of self-adaptation. Currently, only few studies have been published on the dependability V&V of the self-adaptation strategies for SCPS (i.e. only 11 results are returned with the keywords ((TS="Cyber Physical system" or TS="industry 4.0") and (TS="self-adaptive" or TS="self-adaptation" or TS="self-adapting")) and (TS="safety" or TS="reliability" or TS="dependability"))) on web of science as of 2018-8-30. Among them, two papers are about "security" and one is patent). As the environment changes dynamical, the status information of cyber space and physical space has strong timeliness, SCPS should support **real-time self-adaptation** (self-management). However, few publications on real-time self-adaptive CPS are available. The investigation in this section will not be limited on CPS domain. The dependability of self-adaptation mainly involves two aspects, which are 1) the safety and fitness of self-adaptation decision and 2) the dependability of decision process. In this section, we first briefly introduce the state of the art of self-adaptation methods. In the subsection 5.2, we discuss the solutions to make safe self-adaptation decisions. Then, we introduce the methods to guarantee the safety of decision processing at runtime in subsection 5.3 and the dependability of realtime decision processing in the subsection 5.4. Finally, a brief summary is made in subsection 5.5.

5.1 Brief overview of the state of the art self-adaptation for CPS

Self-adaptation is a key technology to build the complex autonomic computing system, and we believe that self-adaptation will be the technological trend for CPS. However, the strategies for SCPS is just at the beginning, a few researches on SCPS have been published due to 2018-8-30 (as seen in the last three rows in Table 4 in the Appendix A). Theoretically, self-adaptation can be abstracted as dynamic optimization problems (DOPs). From the view of the main application levels in dynamic environments, the centralized self-adaptive algorithms can be classified into three types: applied at the metaheuristic level, applied at the mechanism for DOPs level, and applied at the combination level; and the conclusion of the survey shows that the self-adaptation algorithms with at combination level are significantly better than others [35]. However, the real-time constraint and decentralized self-adaptive algorithms haven't been analyzed in this survey. More detailed introduction of the patterns of self-adaptation strategies are recommended to [74, 158-159], and the survey of swarm intelligence algorithm are presented in [160-162]. The statistics of literatures on self-adaptive CPS are investigated in [74, 163]. As detailed survey on the self-adaptation algorithms is beyond the scope of this paper, we just briefly introduce the self-adaptation strategies for SCPS in this subsection.

As shown in Table 1, the current self-adaptation solutions include *rules based methods* [46], and domain dedicated *knowledge* [58] or *ontology* [164] based methods. To develop these methods, it

generally needs to spend long time preparing the rule/knowledge base. And evidence/case based reasoning methods are generally interpreted as a supplementary solution. As SCPS has to face various situations, it generally needs a large rule/knowledge base, which is a big resource burden to SCPS, especially to the battery supported subsystems. To overcome the problem, “AI” methods are proposed for different subsystems, such as *reinforcement learning* based methods for actuators like robots [165-167], *swarm intelligence* based methods for embedded subsystems [13, 162] and *machine learning (ML)* based methods for big data analysis [168].

“AI” based methods, especially the statistical learning methods, need massive training data and huge computation resources. Currently, most of exploratory studies focus on big data driven decision making. E.g. Wang H.T. et al. predicted the influence changes of facilities over dynamic vehicles with Bayes method and trajectory-based Markov chain model [169]. Chen Y.Y et al. introduced an objects searching system for IoT with context-aware hidden Markov model and ontology methods [164]. Martin M. et al. compared the effect of deep learning and four multi-class classifiers which include multiclass neural network, multiclass decision jungle, multiclass logistic regression and multiclass decision forest on data processing in Industry 4.0 [168]. Kush R.V. et al. redefined safety based on the terms of risk, epistemic uncertainty and harm for statistical machine learning [170]. Alexander Maier presented an appropriate passive online learning algorithm based on timed automaton for cyber-physical production systems [171]. To get good results and to avoid overfitting, mass of high quality training evidences/data are needed [172]. Currently, these high quality evidences have been carefully pre-filtered and (manually) marked. And the current applications are generally dedicated to several fixed scenarios. In the real world SCPS, elaborate data is generally not available. Meanwhile the amount of scenarios are very large and their bounds are ambiguous. In addition, the long training time and the high overhead limit the online applications on tiny sensors and actuators. The statistical results are generally lack of personalization, which may not fit well to the individual nodes in some cases, while CPS applications are location-aware and object-aware. However, no technology can simultaneously learn from both statistical data and historical data.

The research of “AI” based self-adaptation decision making is far from mature, enormous technical challenges have to be overcome [19, 163, 173], especially the safety and correctness verification problem. As SCPS is safety critical, it is important to guarantee the C&D of self-adaptation decision. However, most of “AI” methods are weak in interpretability. No solution can analyze the effect of each operation of “AI” decisions, which makes it difficult to verify the dependability of “AI” methods. What’s worse, it takes long time to complete the whole self-adaptation loop (from sensing to making decisions and taking actions). The lagging decisions maybe no longer fit any more to the new situations. *Proactive and latency-aware self-adaptation* [174] is the only alternative solution. However, forecasting is a harder problem, no (“AI”) methods can effectively predict the future statuses of a complex system, such as the future environment, and the SCPS itself. Therefore, “AI” based self-adaptation is not mature for the practical SCPS, but it can be an assistive technology.

5.2 Improve the fitness and safety of the prophetic self-adaptation decisions and strategies

Though many self-adaptation models and strategies have been proposed [17, 19], the safety of self-adaptation is still an open issue. Considering great values of SCPS, it urgently needs methods to guarantee the safety of self-adaptation decision, especially the prophetic self-adaptation decisions [175-177]. Roughly speaking, we can estimate the fitness/safety of self-adaptation strategies from two perspectives at the decision making period. One is calculating the safety with a fitness function and make safety aware self-adaptation decision; another is verifying the safety requirements with V&V methods.

5.2.1 Safety aware self-adaptation decision making

Generally, the safety can be quantified with the loss/risk if the decision doesn't well fit the target context. Hence, we can apply risk assessment methods to estimate the safety of self-adaptation decisions. Massive methods and models are proposed in various domain to estimate the risk/loss if the decision fails [178]. However, risk assessment is still an open issue, because various factors affect the safety of a decision, such as the huge amount of influential factors, the incomplete evidence and the error of data etc. The traditional risk assessment solutions include rules based methods, domain dedicated knowledge and ontology based methods [179], which are highly dependent upon the priori domain knowledge.

To overcome this issue, "AI" based solutions can be applied. Similar to generic ML methods, we can define the risk functions as the (property) loss function. Mathematically, given the joint random variables X (features) and Y (labels) with probability density function $f_{X,Y}(x,y)$, and the mapping functions $h \in H: X \rightarrow Y$ (hypothesis), then we have the normal loss function $L: H \times Y \rightarrow \mathfrak{R}$, the property loss function $C: H \times L \rightarrow \mathfrak{R}$, and the expected value of loss/risk $R(H): \iint_{x,y} C((y, L(h(x), y))) f_{X,Y}(x,y) dy dx$, where $L(h(x), y)$ is the discrepancy between the hypothesis $h(x)$ and real value y and $C(y, L)$ is the property loss if we accept the hypothesis $h(y)$ and process the corresponding activities. The safety-aware ML methods search the best H with minimized $R(H)$ and acceptable size $|H|$.

As a well-known conclusion, the accuracy of ML solution depends on the quality of training data. The main challenge of safety-aware ML methods is how to collect large volume and high quality training data, especially in the cases that the failure data is scarce and the accurate property losses are difficult to evaluate. Moreover, the prophetic self-adaptation decisions are made based on the predicted statuses. Besides the inaccurate prediction, it is also impossible to enumerate all statuses and collect the corresponding priori knowledge of loss (during training). Hence, in most of cases, H is incomplete and C is inaccurate. Priori knowledge integrated with ML methods may be a potential solution to reason in the cases data, and non-strictly quantitative loss function $C: H \times L \rightarrow \mathbb{N}$ can be applied as a tradeoff solution to build a rough mapping

5.2.2 The safety V&V methods at design period and at decision making period

The research on V&V of self-adaptive system is a popular domain. Currently, many related surveys have been published. For example, a detailed statistical analysis on formal methods for self-adaptive systems is shown in [180]. The statistical result shows that formal modeling languages with property specification is the first modelling choice and reasoning is the main research technology. Formal V&V methods are mainly used to verify efficiency, reliability, functionality and flexibility. Embedded system and service based system are the top 2 application domains. Another survey comprehensively discuss the runtime verification for service-oriented systems from five categories: the logic and calculus oriented approaches, runtime workflow monitoring, state-based conformity assessment, aspect-oriented verification and SLA-driven compliance [181]. For each solution, the authors introduced the framework from 8 selected characteristics, which include monitor, monitoring process, formal specification language, development language, monitoring data, realization mechanism, properties of interest and service composition. Then the authors conclude that developing dependable service-oriented systems need elaborately design of runtime verification process and self-healing and self-adaptation mechanisms.

Compared to the generic self-adaptive system, SCPS needs to support real-time self-adaptation and the time for V&V is limited. In some cases, these procedures should be executed in realtime. Besides the generic characteristics, the V&V of realtime self-adaptation should also verify the time related constraints, such as the BCET and WCTE, the convergence rate of strategies and recovery time of failures, as well as the time cost of verification. Currently, the V&V of real-time self-adaptation is a new topic. Only 18 papers (patents are excluded) are returned on web of science with the following keywords ((TS="real-time" and (TS="evaluation" OR TS="validation" OR

TS="simulation")) and (TS="self-adaptive" or TS="self-adaptation" or TS="self-adapting") and (TS="safety" or TS="reliability" or TS="dependability") not TS="security").

To automatically verify the safety of self-adaption decision, SCPS should firstly specify the requirements of decision (processing), and then model the coming scenarios, then evaluate the fitness of decisions and hazards of decision processing under temporal constraints. To model the different decision requirements in different transportation scenarios, a self-adaptive requirements specification is proposed to instruct the behavior of intelligent transportation systems [182]. According to the given example, the specification includes both functional and nonfunctional requirements, and its format is similar to software development specification. The authors also mentioned that a KAOS approach to model the adaptation goals. In V&V domain, aiming at the temporal dependent correctness, a zone-based Time Basic Petri nets specification formalism is designed to model the timed self-adaptation behavior and to verify the timed robustness properties of self-healing [183]. The framework can divide the Petri nets into several zones to simplify verification of intra-zone properties, and apply inter-zone evaluation the global properties. Zone is a divide-and-conquer solution, which is a useful methodology to solve the state space explosion issue (MQ2). Another Petri Net based validation approach is proposed for speed synchronization racing car. This approach rewrites the safety properties and the Petri Net model with Z3 SMT solver, and automatically proves the safety properties in both design and implementation level [184]. To verify the time-bounded decision process, M.B et al. formalized the semantics of the adaptation logic of the component called OLIVE with constraint LTL over clocks and verified the properties with SMT-solver. The authors analyzed the upper bound of the temporal cost of both on-line and off-line verification. Further, the authors pointed out the need of ad hoc approaches to perform on-line verification and discouraged the use of general formalisms [185]. Generally speaking, these formal methods extend the generic solutions by attaching the temporal constraints to self-adaptation model, and evaluate the safety of realtime adaptation by verifying the correctness and time bound of operations.

As the architecture plays an important role in self-adaptation, a comprehensive solution should not only verify the self-adaptation strategies, but also evaluate the fitness of applied architecture/topology and strategies, and the synergy between them. As a solution for MQ3, a systematical approach is proposed to deal with the self-adaptation under uncertainties and topology changes. The authors extended the MAPE-K loop with a rule based structure knowledge base and a goal based requirements knowledge base. On these basis, the authors also proposed a formal timed adaptation logic to build adaptation rules and a meta-adaption layer to verify the adaptation rules with SystemC and UPPAAL timed automata. The meta-adaption layer can evaluate the effect of previous adaptations and learn adaptation rules based on runtime models [186]. Ilias G. et al. proposed a goal-oriented design method for self-adaptive CPS, which is called IRM-SA (Invariant Refinement Method for Self-Adaptation). The IRM-SA traces the requirements between system and the distinct situations of environment, and supports the verification of the dependability of self-adaptation. To deal with operational uncertainty, the authors also introduced a predictive monitoring mechanism for ABSA system in their proof-of-concept demonstration [99].

Simulation based V&V methods are a useful solution to overcome the lack of possibility distributions (for MQ1). Compared to formal methods, it is easier to analyze the synergy between self-adaptation strategies and self-adaptive architecture. A stochastic game based method is presented to analyze the proactive latency-aware self-adaptation. The authors built a formal probabilistic model and verified the adaptation behavior with discrete event simulation [187]. Paolo A et al. provided a formal rule based validation method for service-oriented applications and validated the behavior by scenario-based simulation [143]. Youngil K et al. introduced a vehicle-driver closed-loop simulation method to verify the self-adaptation algorithm for integrated speed and steering control with the CarSim and MATLAB/Simulink [188]. A rule based self-adaptation algorithm are proposed for traffic light control, the authors analyzed the safety and liveness with Simulink [189]. Simulation based methods can evaluate the safety by checking predefined

rules/thresholds but also the statistics of unacceptable results. Obviously, simulation based V&V methods can be regarded as a part of MDE solution, which suffer all the challenges that have been introduced in the subsection 4.3. Considering the computation complexity of simulation, it is only possible to evaluate some subsystems in some specified scenarios.

Moreover, uncertainty is the main challenge of the V&V of the (prophetic) self-adaptation decision for future scenarios [190]. As a possible solution, we could integrate the risk assessment methods in to V&V methods, and estimate the risk of mismatching between (prophetic) self-adaptation decisions and scenarios, especially the risk of mismatching caused by the environment changes during decision processing (for **RQ3** and **RQ4**). To deal with the uncertainties, one native idea is to determine the scope of self-adaptation strategies at the design period (with simulation methods), i.e. Building the set of all applicable situations where the fitness of self-adaptation is large than a given threshold or the $R(H)$ is less than a threshold. Then SCPS estimates the variation range of future scenarios at runtime and verify the safety by checking the coverage of two scopes

5.2.3 Brief summary and discussion

The research on the safety V&V of realtime self-adaptation is just at the beginning. To develop safety V&V for SCPS, we should consider not only the issues in generic self-adaptive system, but also the temporal constraints, the adaptability of architecture/topology and the uncertainties, as well as the process time of V&V methods. Currently, formal solution and simulation are the two main means applied in the safety V&V. These two means can't verify "AI" based self-adaptation. To the best of our knowledge, there are only three conceptual discussions on the safety V&V for "AI" solution. Sanjit A. et al. introduced a concept of trustworthy machine learning and argued to use end-to-end specifications and specifications mining to guarantee the safety of "AI" [175,176]. Zolt'an T. et al. introduced an reinforcement learning based event-learning framework to improve the communication and control of SCPS, then discussed the problems of verification from context and goal [167]. From our point of view, we might be able to apply safetycases@run.time, simulation based methods and even "AI" based verification method to verify the safety of "AI" based self-adaptation. More detailed concept will be introduced in the subsection 5.4.1 and 7.4. All in all, with the application of "AI" based self-adaptation, it urgently needs to study the safety V&V methods for "AI" based self-adaptation

5.3 Guarantee the safety of decision processing with model@run.time methods

As argued earlier, SCPS should make prophetic self-adaptation decision to overcome the long delay. However, as future is full of uncertainties, future prediction is still an unsolved problem, let alone reasoning the future statuses and making proper decision. One direction to reduce uncertainty is making short-term self-adaptation decision. However, the shortest term depends on the minimum delay from sensing, making decision to taking actions, which is limited by current physics technologies. Another is gradually refining the decision with model@run.time methods at runtime. Though we can't predict the future, we can specify the applicable context for each self-adaptation decision and let the actuators to decide to process the decision or not. Model@run.time is a systematic solution for prophetic self-adaptation, it can significantly reduce the unnecessary uncertainties [191]. Moreover, with Model@run.time methods, SCPS can decompose the V&V procedure, all involved subsystems can verify the sub-decisions and refine them with newest observation under the guideline of requirement specification, which is a promising solution for realtime V&V.

Due to the potential capability to reduce the uncertainties, model@run.time methods attract more and more attentions. The survey [192] introduced the model@run.time methods for open adaptive systems (OAS) and analyzed the state-of-the-art safety assurance solutions and challenges of model@run.time. The model@run.time methods were classified into 4 types:

safetycertificates@run.time, safetycases@run.time, V&V model@run.time, and hazard analysis and risk assessment@run.time, where safetycase@run.time is a formalized, modular safety case, which can be interpreted and adapted at runtime. The safetycase@run.time methods attach the complete arguments to interpret the requirements of safe statues and safe situations. A good safety case model has good quality of C&C. The top-level safety goals can be decomposed into the detailed requirements for each subsystem. Therefore, the subsystems can dynamically check the safety goals with the safety requirements at runtime.

Mario T. et al. argued that "safetycases@run.time appears to be sufficient to support the assurance of a wide range of application scenarios of OAS (Open Adaptive Systems) in safety-critical applications" [192]. However, in our point of view, safetycases@run.time is not enough to guarantee the safety of SCPS. As analyzed in the subsection 3.3.1, multi-term self-adaptation loop is necessary to overcome the communication delay and the shortage of resources. Requirements@run.time (with compositional decision) is necessary to instruct the cooperation among distributed subsystems, and requirements@run.time can also reduce the resources budget and uncertainties. Moreover, compared to OAS, SCPS has larger scale and more critical constrains on realtime. Model@run.time of SCPS should not only consider the all challenges that OAS have to face (as seen in the subsection 6.2 of the survey [192]), but also the realtime constraints, especially the performance of interpretation.

However, current model@run.time approaches are fragmented and dedicated to limited scenarios, no integrated safety assurance framework is available [192], which can't well describe the multi-level architecture and model the dynamic behavior of SCPS. To model and verify continuous scenarios of SCPS, model@run.time solution should dynamically adjust the rules and cases. And to overcome the limited resources of embedded subsystem, we need to develop compositional models and rules, take full advantage of the hybrid multi-role self-adaptive architecture, and build multi-level databases to cache the rules and cases. Moreover, the DSS should be able to recognize the scenarios and subsystems should be able to dynamically rebuild models to fit the scenarios.

As a kind of safety V&V method, the model@run.time solutions also suffer from the issues introduced in subsection 5.2.2. And similarly, the research on realtime model@run.time is rare. No result was found in the web of science with ($TS="model@run.time"$ and $TS="real-time"$ and ($TS="safety"$ or $TS="reliability"$ or $TS="dependability"$)), and only 26 papers are returned with the query "(+real-time +self-adaptation +model@run.time) AND keywords.author.keyword: (+safety)". With further review, there are only 2 concept discussions [193-194] and 2 proposals [195-196] belong to our topic. The two proposals take into account the deadline of adaptation actions [195] and the WCET of subsystems' actions [196], and try to guarantee the safety of self-adaptation on timeliness.

5.4 Guaranteeing the dependability of real-time self-adaptation (decision process)

SCPS is a large scale (geographically) distributed system. The self-adaptation loop takes time. Meanwhile, the cooperation between subsystems suffers from various uncertainties, i.e. **RQ2** and **RQ4**. To guarantee safety of self-adaptation, SCPS should decompose the decisions and send the sub-decisions to the proper subsystems. Thereafter the subsystems should cooperate with each other (with the coordination of local DSS) to guarantee the timeliness and dependability of sub-decisions process. In addition, SCPS should apply multi-level dependability aware scheduling, and be ready to take remedial measures when subsystems fail or the decision isn't suitable any more. In this subsection, we assume that SCPS have made correct and safe prophetic self-adaptation decisions, and focus on the dependability of self-adaptive decision processing.

5.3.1 Guarantee the dependability of self-adaptation with multi-object optimization methods

SCPS should process the decision at the right time and at the proper speed with as few as resources as possible. Various factors can affect the progress of decision processing, such as, the unexpected failures and the uncontrollable jitters. To complete the decisions timely, efficiently and spontaneously, SCPS should take into account various runtime requirements, generate the optimal

decision process plan, then select the proper executors, arrange redundant subsystems and set backup plans in case of failures. The problem of long-term plan making can simplify as the generic multi-object optimization problem (MOOP). Current optimization objects for SCPS include guaranteeing the realtime/deadline/timeliness [197-198] resource performance [199], dependability [73] and other functional objects[200]. Massive algorithms have been proposed for MOOP, such as evolution algorithms [201], an natural inspired algorithms (i.e. Ant colony and swarm optimization)[202]. As MOOP is a well surveyed topic, here, we will not analyze these algorithms in detail.

In case of short-term adaptation/scheduling, SCPS should schedule several self-adaptation procedures simultaneously, i.e. for self-driving cars, system should adjust the speed to avoid the various collisions, plan its path to avoid congestion, and process the commands from passengers. The short-term self-adaptation/scheduling for SCPS is dynamic multi-object optimization problems (DMOOP) [35]. Due to the realtime constrain, it generally could not find the optimum solution for DMOOP in time. SCPS has to track the changing optimum with iteration. Moreover, the decisions process involves massive subsystems, whose failure rate is no longer ignorable, especially the activities should be stated at right time and finished in time. To archive dependability aware scheduling is necessary, SCPS should take into account the states of system and environment [203], fault tolerance scheduling [89] and energy aware scheduling [204]. Moreover, most DOP algorithms are designed for centralized applications [35]. For distributed actuators, we need decentralized self-adaptation algorithms to coordinate interaction. A multi-threading multi-agent scheduling was proposed to search the efficient and fast solution of complex problems in realtime featuring rapid dynamic changes and uncertainty [205]. To process dependably, these algorithms should take advantage of design diversities, arrange several (heterogeneous) redundant subsystems in safety-critical scenario.

5.3.2 Guaranteeing the dependability with goal/contract based on decomposable self-adaptation decision (the requirements@run.time approach)

SCPS involves multi-level scheduling [47, 62], the scheduling strategies at each level have different optimization goals and different constraints, some of them maybe even conflicting. In current centralized architecture, subsystems generally know nothing about the global goals and constraints, they just try best to response the scheduling commands. As a result, subsystems could not solve the multi-object optimization problems under their own contexts, and could not verify the non-functional requirements of commands (or sub-decisions). To solve this issue, requirements@run.time solution sends the sub-decisions with the corresponding requirements called contracts (as seen in Figure 7). With contracts, subsystems can know the global optimization goals and slightly adjust the plans to better fit the local contexts.

Fallowing this idea, JIANG et al. first formally defined an event-based scheduling policy and proposed a decentralized scheduling method, this method can automatically decompose the scheduling policies into atomic scheduling policies and analyze those policies independently [206]. Ilias G. proposed a goal driven decentralized self-adaptation solution as a proof of concept of IRM-SA [99]. IRM-SA allows capturing the compliance of design decisions with the overall system goals and requirements; and subsystem could verify the goals and requirements at run-time. Based on formally specifying requirements, Carlo G. proposed a run-time efficient parametric model checking solution to verify the dependability at run-time [130]. As a nonfunctional requirement, we also proposed run-time dependability decomposition and subsystem composition patterns with WCET constraints [73]. The key of goal/contract based on self-adaptation is improving the C&C of subsystems and decisions (as seen in the subsection 3.3.2). Only in this way can subsystems be composed correctly and dependability.

5.5 Brief summary and discussion of dependable self-adaptation

The self-adaptation of SCPS needs the interplay between architecture and strategies. Hence, to guarantee the safety of self-adaptation, it needs to co-verify the behavior of architecture and strategies. And the safety V&V methods should be applied at both design period and runtime to overcome the resource limitations, the uncertainties and the realtime challenge. Currently, formal methods and simulation methods are applied to evaluate the correctness and safety of self-adaptation, and check the satisfiability, reachability and consistency of self-adaptation behavior in different scenarios. However, these methods generally is dedicated to specified scenarios, which can't continuously verify the safety of SCPS. Moreover, it is still lack of safety V&V method for "AI" based self-adaptation. As SCPS contains multi-level self-adaptation loops, it is necessary to apply some methods, like model@run.time and goal/contract based solutions to coordinate the self-adaptation at different level. We believe model@run.time is a useful solution for SCPS to reduce the uncertainty of V&V and achieve realtime safety evaluation.

6. Self-healing solution for SCPS

Dependable SCPS needs self-healing solutions to keep health, prevent failures and simplify the maintenance. However, self-healing methods will introduce extra complexity, which is harmful to achieve high dependability. To overcome the challenges of **RQ1** to **RQ6**, we should **use simplicity to control complexity** [29], elaborately employ both traditional methods and modern methods to reduce the their side effects.

Several similar surveys on self-healing/maintenance have already been published. Santiago R.Z et al. first reviewed the existing maintenance principles for linear complex systems (LCS); through analyzing the similarities and differences between LCS and CPS, the authors gave the non-applicable, exportable and adaptation required principles for the CPS separately [207]. According to the concept of maintenance 4.0 in [208], modern maintenance are mainly focused on self-healing, especially failure prediction.

In this section, we focus on the technologies of self-healing for SCPS. We first briefly summarized the traditional dependability methods applied in SCPS, and then focus on the modern methods of runtime self-healing. Following the classification of the means to achieve dependability, our investigation is organized into 3 parts: **1)** fault tolerance, **2)** fault prediction (forecasting) and prevention, **3)** maintenance. The limitation of each methods are also stated by considering of the characteristics of SCPS.

6.1 Traditional solutions to improve the dependability

The theory and traditional means of dependability [103-110] have been well-studied for a long time. As the investigation illustrated in Table 1, the traditional dependability methods are still the mainstream choice to guarantee the dependability of SCPS, which get impressive results, especially in guaranteeing the dependability of infrastructures. As SCPS contains abound of subsystems (especially the cloud based SCPS and MAS based SCPS), *subsystem level redundancy* is a natural solution to improve system dependability. For cloud based SCPS, VM based resources isolation can significantly simply the fault isolation approaches [45, 47, 49]. *Rule* or *knowledge* based fault diagnosis and reasoning show good results in fixed or rarely changed environment [59]. Expert system is a systematic dependability solution [209]. Dependability *middleware* is also an alternative solution for heterogeneous subsystems, which can provide consistent behavior [36].

As SCPS is a holistic system, it need to apply dependability means at all levels and all subsystems and protect the SCPS from different perspectives. Focusing on dependable hardware engineering, Sparsh M. et al. provided a comprehensive survey on reliability techniques for microarchitecture design (such as processor registers, cache and main memory [210]. From the time perspective, Z.W. Gao et al. studied the real-time signal based *fault diagnosis* techniques and *fault-tolerate* techniques in control domain [211]. From the data-centric perspective, Thaha M. et al. considered 7 types of faults into account including offset fault, gain fault, stuck-at fault, out of

bounds fault, spike faults, data loss faults and aggregation/fusion error, and performed a qualitative comparison of the latest fault detection researches including centralized, distributed and hybrid algorithms in WSN domain, and analyzed the shortcomings, advantages of those algorithms [212]. Focusing in WSN domain, Samira C. et al. reviewed the *fault tolerance methods* at different WSN levels and classified these methods into 4 categories: power management, flow management, data management, and coverage/connectivity [213]. Massive methods are available to guarantee the dependability of static architecture and predefined software operations. The systemic introduction to dependability/reliability engineering are recommended to the books [103-105]. It should be mentioned that all introduced dependability management belongs to the active fault handling measures. In some cases, “**let it crash**” maybe the best solution in some cases, because fault diagnosis may take too much time and dependability means may cost too much.

6.2 Modern methods for fault tolerance

Fault tolerance is a key technology to guarantee the dependability at runtime. Through design diversity, SCPS contains numerous heterogeneous subsystems with isomorphic (but different) software, which provides an ideal platform for applying the temporal-spatial redundancy mechanism at the subsystems level. Following the general process flow of fault tolerance, we organize the investigation into 3 parts: fault detection/diagnosis, fault isolation and fault recovery/tolerance.

6.2.1 Data driven fault detection/diagnosis

Yang et al. focused on the fault of context inconsistency and surveyed the improvement techniques and the challenges of deploying dependable pervasive computing systems from the fields of context management, fault detection, and uncertainty handling [214]. The authors classified the detection methods into 1) *statistical analysis-based detection*, 2) *pattern matching-based detection*, 3) *coverage criteria-based detection*, 4) *simulation based detection*. Yet most surveyed papers just detect the inconsistency between the function context and the environment, few of them detect/verify the inconsistency between the dependability management context and the status of system and environment. As mentioned in subsection 2.4 (the schema (b) of self-healing), self-healing is also a common service of the system, which also needs self-healing. Moreover, the (proactive) self-healing operation may be wrong, it needs V&V methods to check the fitness. Verifying and detecting the self-healing service can decrease the false positive rate of self-healing. Moreover, we also need to optimize the release time of self-healing to guarantee that the healing operations take effect before the system crash.

From the academic view, the survey investigated the knowledge-based fault diagnosis and hybrid/active diagnosis [211]. Furthermore, the quantitative knowledge-based fault diagnosis can be divided into 1) *statistical-analysis data-driven fault diagnosis*, 2) *nonstatistical-analysis data-driven fault diagnosis*, and 3) *joint data-driven fault diagnosis*. Similar to all “AI” based method, the training data is the key to achieve the highest quality diagnosis result. However, (high quality) failure data is scarce because of the difficulty of data collecting and small failure rate. Moreover, data driven diagnose is good at recognizing the abnormal (unexpected) symptoms, but not identifying real failure causes. As abnormal symptoms may be caused by normal behaviors, i.e. stochastic resource competition, or just in unrecorded small probability situations, without additional reasoning methods, the data-driven solutions will generate the massive false alarms and increase the false positive rate.

As knowledge-based fault diagnosis needs huge resources (memory and storages), which is generally applied in the cloud system (DSS). For sensors and actuators, light-weight detection/diagnosis methods are the mainstream solution, e.g. heartbeat methods, WCET based detection, and threshold/event/state based methods. Modern methods for embedded system include: 1) *Model based fault detection* [215-216], yet the flexibility is limited. 2) *Probabilistic fault detection and isolation* [217], which may increase the rate of False Negative. 3) *Cooperative fault detection*

[218], through exchanging (checking) information between neighbors. 4) *Distributed fault detection* [219], based on the statistical result of the neighbors' data. 5) *Machine learning based fault detection* [220–222], these methods are generally applied in the data center, and the performance depends on the quality of collected data and the complexity of tested system.

Similar to WSN, the faults of SCPS are local in the network. Hence, we learn the fault detection strategies in WSN [212], and develop suitable strategies for SCPS. Compared to WSN, SCPS is location-aware and individual-sensitive, the data in each location is unique and may be irreplaceable to decision making, using the approximate value from neighbors will decrease the fitness of decision and increase the risk of failure (for **RQ2**). It needs to comprehensively apply statistical based, time series based and domain knowledge based validation solution to analyze the reliability of data.

6.2.2 Virtualization based fault isolation

Through isolating the resource and execution environment between applications, virtualization can isolate the faults at the subsystems level and naturally prevent the propagation of faults. For DSS, we can apply virtual machines (VM) or container to isolate applications. For sensors and actuators, faults can be isolated by separated hardware, middleware [223] and container [224]. Virtualization can also be applied to improve the isolation of networks. A virtual platform called Xebra is proposed to provide isolated communication for SCPS [225]. Both node-level and network-level virtualization are studied for WSN [226], the authors classify the network virtualization into *virtual Network/Overlay-based Solutions*, *cluster-based solutions*, and *hybrid solution based on middleware or VM*. Besides the advantages summarized in subsection 3.2, virtualization is a useful isolation solution and the snapshot of virtualization is also a convenient recovery technology. Meanwhile, migration technology enables to switch the workload from failed hardware and improve the availability of services.

6.2.3 Modern fault recovery/tolerance solution

Though we can apply fault tolerance solutions at different level, one things should be kept in mind about that SCPS suffers the famous Byzantine failure. The Byzantine fault is any fault presenting different symptoms to different observers. As a networked parallel system built with unreliable (wireless) communication channels, Byzantine failure will destroy all fault tolerance solutions by undermining the consensus of observation. Thanks to the virtualization technology, an advanced $2f+1$ replication solution is designed to achieve the Byzantine fault-tolerate [227]. Manuel R. et al. explored the role that diversity plays in defending faults and attacks through qualitative analysis, the result shows that diversity can tolerate the common-mode failures but will introduce undetected failures [68]. The effects (robustness) of redundant connections are studied through the simulation of six kinds of topology [228]. To tradeoff between the cost and reliability, a formal specification of spatial and temporal redundancy based reliability is proposed; and a framework is designed to determine the maximum reliability with a recursive formulation [229].

6.2.4 Brief summary of fault tolerance methods

For SCPS, as all the redundant components suffer the same environment, redundancy with the same subsystems fails to deal with interference from the physical world. Diversity design with heterogeneous subsystems is the solution to tolerate these failures. To build a dependable SCPS without introducing too much complexity, we should take advantage of self-adaptive architecture by systematically adopting the spatial-temporal redundancy solutions and isolation at different levels. Following this idea, Cui. et al. introduced a cross-layer fault tolerance method based on self-adaptive architecture [230]. Moreover, we should also take into account the time cost of fault diagnosis and recovery, avoid resource competition with normal functions.

6.3 Modern methods for fault prediction and prevention

The ideal self-healing solution is predicting the faults and then preventing the failures. The fault prediction and prevention methods monitor the symptoms or abnormal behaviors, and evaluate the risk of faults, then prevent the failure with proactive measures, i.e. replace the fragile component, or replace with safe actions, or reject (or freeze) the dangerous operation (i.e. reject to accelerate a car at unsafe distance). Fault prediction and prevention have been widely applied in IoT domain. A fault prevention framework is proposed, which consists of monitoring, fault-prediction, and adaptation mechanisms [231]. To prevent selecting the unhealthy node as cluster heads, a fault-prevention clustering protocol is designed, which includes failure prediction, cost evaluation, and clustering optimization [232]. Deshan C. et. al proposed proactive systematic self-adaptation solution based on fault prediction and prevention to improve the reliability of mission-critical, embedded, and mobile software [233].

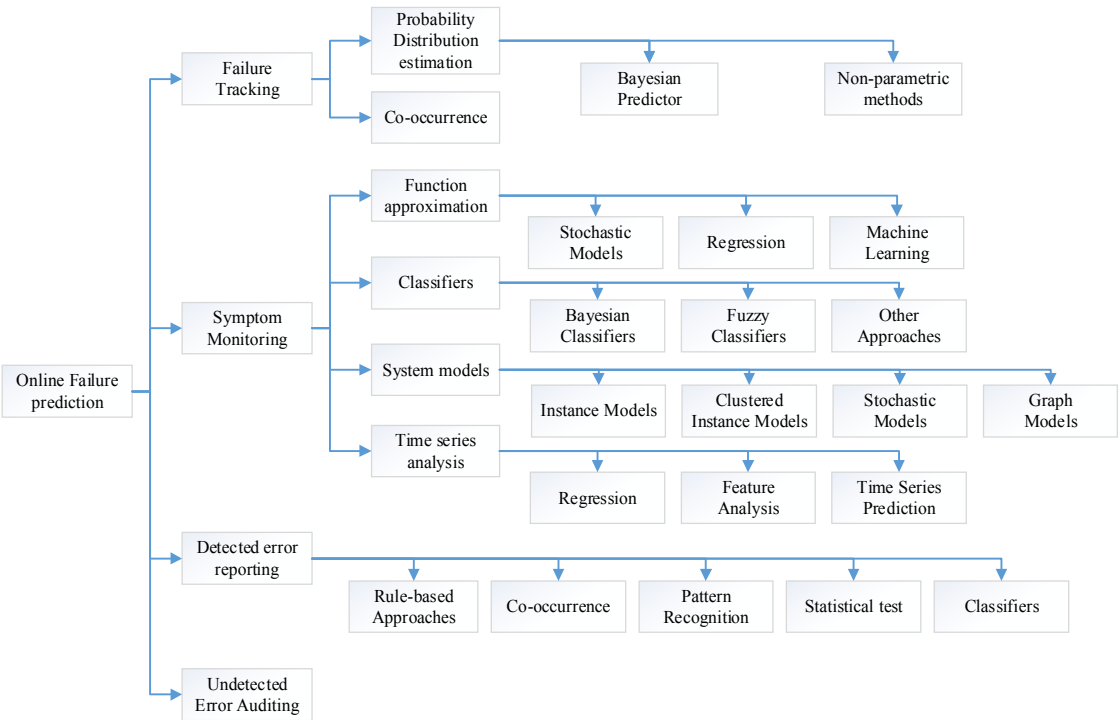


Figure 11. A taxonomy for online failure prediction approaches

Technically, the key of fault prediction is monitoring the leading symptoms, such as the received signal strength indicator (RSSI) to battery power depletion [234] and the energy consuming state to estimate the remaining lifetime of battery [235]. *A priori causal knowledge of failures is the selection guidance of leading symptoms, which can be gained from the traditional dependability solution, i.e. expert system [236] or the reliability models [237] (i.e. AZOP and FMEA).* Based on failure propagation models, Claudia P et al. used the failure of error sources component as the leading symptoms to predict the failure of error sink component, and guild the reconfiguration based self-healing [238]. The detail classification of online failure prediction methods is shown in Figure 11 [239].

As mentioned in section 2.3, the self-management loop suffers non-negligible approximate errors. And the probabilistic result for statistics based fault prediction is another serious error source. The prediction results are generally expressed with a probability value whose domain is $[0,1]$, while the execution of prevention and recovery operation is binary $\{0,1\}$ (do or not). For SCPS, mapping from $[0,1]$ to $\{0,1\}$ is incautious and risky. Proactive prevention may take unnecessary, even unsuitable, operations which increases the delay and wastes resources, and even introduces new errors to system. Through causal inference, *model based* or *knowledge based failure*

prediction & prevention may relax this issue. Furthermore, considering the complex error sources, current failure prediction methods are still not powerful enough to diagnose the failures in real-time. *Long-term* and *short-term failure prediction* should be selectively implemented at different levels of SCPS according to the accessibility to the symptom set. Current prediction solution is monitoring one specified symptom for one (kind of) failure. Considering the complex of failures, more efficient symptoms should be researched urgently, thus we can use fewer symptoms to predict more failures, which can significantly simplify the failure prediction. What's more, the accuracy of *prophetic failure prediction* should be improved, especially in unexpected situations. Moreover, as current fault prevention method rarely consider their influence on normal behaviors and neighbor subsystems, we should verify the safety of these prevention operations and use multi-object dependability aware scheduling/arrangement (as seen in the subsection 5.4.1) to coordinate failure prevention operations and self-adaptations, to achieve side-effect free self-healing (RQ4).

6.4 Simplify the manual maintenance

As an inference of the halting problem, self-healing solution can detect and heal all failures automatically. Moreover, some accumulated failures can't be recovered without external help, i.e. wearing out of hardware, and the exhausted battery. Therefore, manual maintenance is still needed. As SCPS contains massive subsystems, it is a big challenge to locate the failed nodes and replace with an identical subsystem. Hence, we should redesign subsystem make it be able to join the SCPS automatically, seamlessly without introducing additional interference (RQ4). Notice that the maintenance here represents recovering the system from the failures, i.e. replacing a broken engine for a self-driving car.

There are three main directions to simplify the maintenance. One is *simplifying the failure identification and location*. Maintainers can locate the failed nodes based on the reports from the neighbor nodes. Another technique is developing assistance programs to improve the efficiency of preventive maintenance. A *condition* and *time* based maintenance model and framework is proposed and *domain knowledge* based decision making for maintenance is highlighted [240]. The Bayesian networks based assistance programs for dependability, risk analysis and maintenance is reviewed in [156]. A linear programming-based model is developed to optimize the repair priorities and a stochastic model is introduced to examine these reparation policies [241]. The third direction is simplifying repairing/replacing. The basic solution is designing the subsystem with high C&C, so the new subsystem can join into the SCPS seamless.

6.5 Brief summary and discussion

To improve the runtime dependability of CPS, comprehensive fault management methods should be elaborately integrated at different levels, and the side effects of self-healing methods should be evaluated at both design period and runtime. The means of developing a dependable SCPS are: **1)** strengthening the reliability of infrastructures (hardware, software and network) by using the traditional methods (for RQ1); **2)** improving the availability of sensor (network) and (network) with subsystem redundancy and reconstruction (for RQ1); **3)** integrating the fault detection/diagnose methods (i.e. FDIR) on sensor nodes and actuator nodes to block error propagation and limit the failure effects (for RQ1); **4)** enhancing the fault isolation at subsystem/application level (i.e. virtualization technology) (for RQ1); **5).** using group intelligence (i.e. DFD, cooperative detection) to check with each other (for RQ1 & RQ2); **6)** applying data-driven methods (i.e. machine learning) on DSS to predict faults (for RQ1 & RQ5); **7)** building knowledge based methods or expert system on all subsystems to evaluate the correctness of prediction and then safety of prevention operations (similar to means for self-adaptation); the knowledge based methods or exporter system can take over control of self-healing system if the big data is

unavailable (for **RQ1 & RQ5**). 8) Simplify manual maintenance and building human-in-loop SCPS (the backup solution for the halting problem).

Self-healing solutions can improve the dependability of the system, but they also increase the complexity and introduce new errors. Finding more efficient symptoms is one solution to reduce the complexity. To develop high performance, low side-effects self-healing solution, we should adopt these principles: 1). **keeping the fault management measures simple and verifiable**; 2). **Taking the side effect of fault management into account** (e.g. time delay, resource budget etc.); 3). If recovery costs are too high then subsystem can just **notify the stakeholders** (other subsystems and managers) and **let it crash safely**.

Current self-healing solutions are not powerful enough for SCPS. We can improve them in these fields: **1) the timing order of events may be wrong (RQ2)**, we can use priori knowledge and cross validation to check the correctness of input data to improve the accuracy of data driven solutions. **2) Environment effects** are not considered in current researches. As environment can directly affect the infrastructures of SCPS, especially the sensors and actuators, it will absolutely change the distribution and rate of faults. Fault prediction methods should take the environment effects into account. **3) Human effects** are also not evaluated. The unpredictable and controllable human behaviors will be the new source of errors. It is urgently necessary to develop the theory to model the human behavior and predict the intention of human.

7. The missing pieces of the technology puzzle and the future directions of SCPS

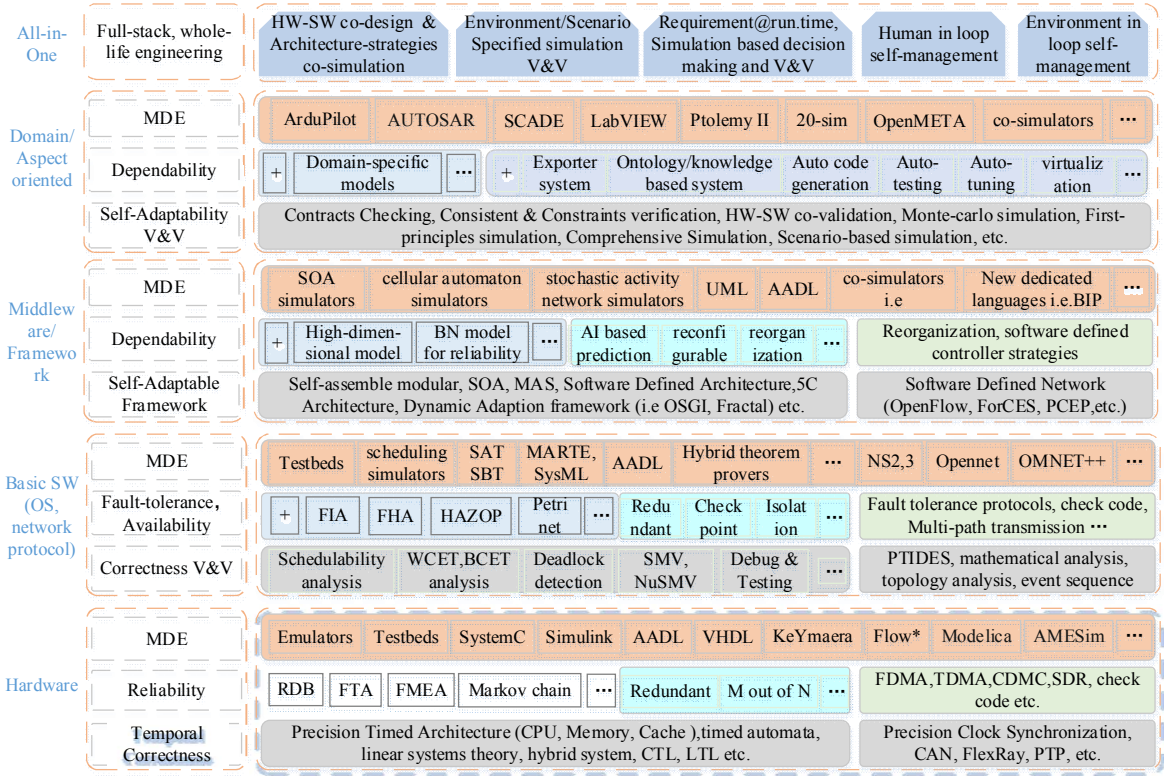
SCPS is a rapid progressing research area involving multidisciplinary technologies. It requires great skills to trade off among the complex requirements and integrate the fragmented technologies together to build an organic system. Moreover, to adapt to the dynamic uncertain environment, SCPS should balance the controllability and autonomy (as seen in the subsection 3.1), automatically make proper self-management decisions with the incomplete, insufficient, even incorrect observations (**RQ2**), and coordinate subsystems to process the decision safely, dependably and timely in the changeable environment. To build dependable SCPS, it needs systematic architecture design, creatively integrate multidisciplinary technologies, and life-cycle V&V and continuous maintenance.

7.1 The available and missing measures

To develop a dependable SCPS and to guarantee its dependability through the whole life cycle, we should comprehensively apply various technologies at different levels. Here, we resume the popular (and potential) methods and frameworks from thousands of papers, classify them from four domains: correctness V&V methods, self-adaptation solutions and V&V methods, dependability and V&V methods, MDE solutions (toolsets), at five main levels: 1) hardware level (nodes and network devices), 2) Basic software level (operating system and network protocols), 3) Middleware and framework, 4) Domain/aspect oriented solutions, 5) the all-in-one solution. Considering the readability, we select part of the results, which are illustrated in Figure 12.

According to the jigsaw of technologies, we can draw eight conclusions. **1)** The researches of precision timing behavior are just at the beginning stage, and most of current researches focus on the hardware and basic software level, few systemic tools or theories are proposed. **2)** The dependability methods can be applied at any level, while the weight of flexibility increases with the rising of level. **3).** Simulation based methods and dynamic logic based models (whose analysis procedure is also simulation based) dominate the V&V of correctness and dependability at the high level. **4)** C&C is a promising solution to improve the flexibility and tame the increasing complexity. **5)** Highly flexible self-adaptive architecture (i.e. ABSA, SDA, and SDN) is the cornerstone of SCPS. **6)** It needs the co-design and cooperation of architecture and strategies to archive maximal and optimal adaptability. **7)** MDE is a holistic solution to tame the complexity and the decrease the risk

1668 of CPS engineering. It should be mentioned that 8) specification is also the useful measure to
1669 guarantee the quality of CPS [242-243].



1670

1671

Figure 12. Available measures for dependable self-management CPS at different levels¹

1672 There are nine missing pieces of the jigsaw of current technologies: 1). theoretically,
1673 infrastructure and tools supporting for timing behavior at high level are urgently needed. 2).
1674 Precision timed wireless network is emerging. 3). Network integrated V&V and large scale
1675 supported toolsets are needed. 4). the coverage and performance of testing and simulation should be
1676 improved. 5). the side effect of dependability management should be continually investigated. 6).
1677 dynamic multi-object optimization strategies are necessary. 7). realtime V&V methods for the
1678 realtime self-adaptation are needed immediately. 8). the research on the cooperation of these
1679 dynamic strategies should be put on the agenda. 9). Safe "AI" methods and related safety V&V
1680 methods are necessary for data driven self-adaptation and fault predication [170].

1681

7.2 Technical challenges and directions

1682

1683

1684

1685

1686

1687

1688

1689

SCPS contains various technologies. However, these technologies are fragmented and unstructured, which could not meet the development and maintenance requirements of SCPS very well. In this section, we briefly summarize the challenges of dependable SCPS development, which are shown in Table 3. In addition, focusing on dependability, we analyze the urgency of related technologies from the perspectives of 1) the maturity of the technology [6, 39], 2) the social expectation and acceptance of technology [23], 3) the degree that other technologies depends on target technology (based on Figure 12). Notice that the challenges of security and traditional technologies are not discussed in this table.

1. MDE is an integration of various technologies, the priority of technologies listed in the figure is theory > category of technologies > language > tool. The frequency is the main selection rule.

Notice: As it's impossible to list all related technologies in one figure, here, we just list some popular methods. Due to our limited knowledge, some other methods may be missed. All suggestions are welcomed to refine the jigsaw.

Table 3. The technical challenges of dependable SCPS

Technical Area	Challenges and Directions	Urgency	Target
HW & SW infrastructure development	Precision timed, real-time HW & SW	High	Timing
	Standardization of subsystem (interfaces)	Medium	C&C
	Low power devices	Medium	Energy
Network communication & management	Precision timed network transmission	High	Timing
	Real-time (wired & wireless) communication	High	Timeliness
	Heterogeneous network management	Medium	Maintainability
Architecture design	Atomic service & subsystem design	Low	C&C
	C&C contract, interoperable subsystems	Medium	Self-*
	Discrete-continuous subsystem integration	Medium	Correctness
	Invariant behavior of integration	High	Correctness
	Theory for dynamic architecture	High	Flexibility
	Design methodology for dependable SCPS	Medium	Complexity
Middleware	FDIR middleware & Node level self-healing	Medium	Dependability
	Light-weight virtualization & migration	Medium	Self-*
	Domain ontology, Knowledge database	Medium	Self-*
	Service discovery & combination	High	Self-*
Consistent spatial-temporal & context cognition	Global reference time for large scale CPS	High	Timing
	Low cost clock synchronization	Medium	Correctness
	Global location reference for mobile CPS	Low	Correctness
	Consistent data and context assurance	High	Correctness
Lifecycle management (self-management)	Manage dynamic & changeable architecture	High	C&D
	Multi-objective (prophetic) adaptation	High	C&D
	Knowledge-driven decision making	High	C&D
	Decision/adaptation safety/evaluation	Medium	Safety
	Situation aware self-healing & notification	High	Dependability
	Causality analysis	High	C&D
Modeling & validation & MDE tools	HMI for human-in-loop CPS	High	Usability, safety
	Dynamic architecture modeling	High	Fidelity
	Multidisciplinary modeling	High	Modeling
	Consistent of model transforming	High	Correctness
	Evaluation the correctness of models	High	Correctness
	Holistic modeling theory or methodology	Medium	Modeling
	Situation based model V&V	Medium	V&V
	MDE toolchains (design, V&V, coding, testing) and life cycle V&V supporting	Medium	Consistency & efficiency
	Discrete-continuous-probability model co-sim	Medium	V&V
	Holistic multidisciplinary simulation	High	V&V
Simulation	Environment-in-loop simulation	Medium	V&V
	Human-in-loop simulation	High	V&V
	Fidelity evaluation	High	Correctness

7.3 Future direction: an concept of all-in-one solution

SCPS should adjust architecture and apply proper self-management strategies to achieve best adaptability and efficiency. Considering the complexity and uncertainties, even we have improved the C&C of models, it is still impossible to completely evaluate the synergy between architecture and self-management strategies during design period. Models@run.time based V&V will be the trend to guarantee the dependability of self-management.

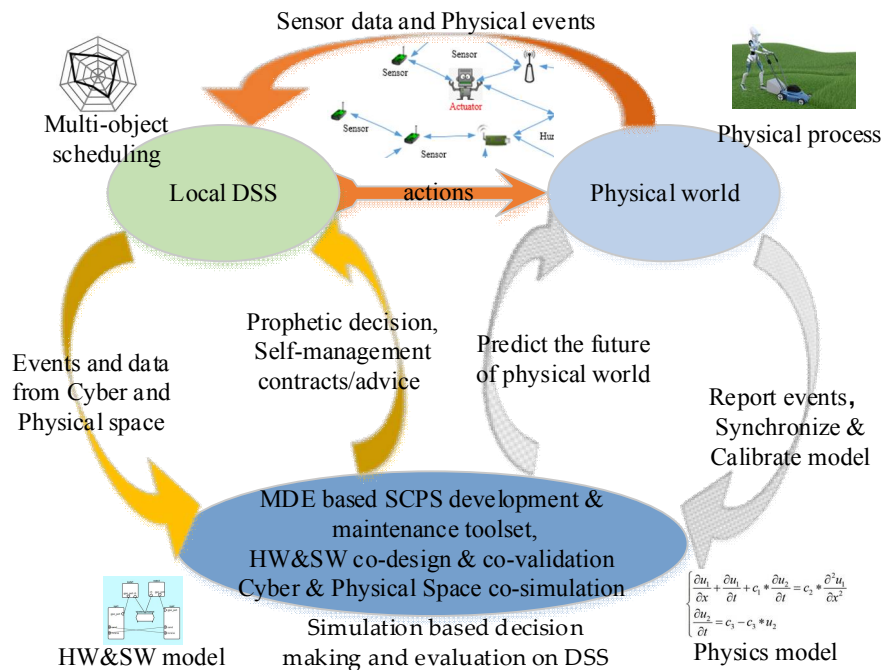


Figure 13. Co-validation and co-simulation through the whole life cycle of CPS

However, current models@run.time approaches are fragmented and dedicated to specified scenarios, which could not well handle the changeable requirements of SCPS and also are unable to verify the system in unexpected situations. To evaluate the proactive self-adaption decisions, the ideal models@run.time methods should be able to predict the future environment $S_p(t + \Delta t)$ without taking any self-adaptation actions; by evaluating the difference between the expected future $S_p^e(t + \Delta t)$ and $S_p(t + \Delta t)$ with the related domain knowledge, SCPS searches the proper self-adaptation actions $\Delta S_p^n(t)$ and self-healing actions $\Delta S_c^m(t)$ based on current statues $S_p^m(t)$, and then generates the related requirement specification, such as timing orders and timeliness of self-management activities, the dependability and robustness requirements of services. To achieve this goal, we can integrate the contract based solutions (as seen in the subsection 3.3.2) and MDE/models@run.time based solution to build a multi-level simulation based self-evaluation solution, which is shown in Figure 13. The conceptual solution integrates a simulation based decision making and evaluation, contract based self-management, and multi-object optimization based (distributed) scheduling. This solution includes three main feedback loops. One loop is between global DSS and local DSS, the second loop is local DSS between sensors or actuators, and the third loop is between sensors, actuators and physical world. Based on the three loops, SCPS can form a complete feedback loop between physical space and cyber space. The global DSS can simulate the both physical space and cyber space (especially the sensors and actuators), then select several best simulation results as the self-management decisions and generate the contract. The local DSS selects the advices/contracts best fitting to its situation, decomposes the contracts based on multi-object optimization, and coordinate subsystems based on multi-object scheduling with

sub-contracts. The sensors and the actuators follow the sub-contracts and process self-management strategies in a decentralized way. This conceptual solution can also provide life-cycle dependability management, and improve the efficiency of development and traceability of management.

As current data analysis technologies can hardly learn the causality from data. Without causality, data driven self-management is full of hazards. While formal models and a priori knowledge naturally contain the causality, which can guarantee the dependability of data driven self-management. With this conceptual solution, we can analyze the causality with the models and generate a holistic view of the causal relationship between events and effects beside on the behavior models, and the causal relationship between error sources and possible failures based on fault propagation models and data flows.

To realize this concept solution, we should develop the MDE toolsets which can model both the cyber space and the physical world and the interplay between them. In addition, with the improvement of MDE toolsets, co-validation and co-simulation programs could be generated from the meta-models automatically. The simulator can evaluate the current situation and even forecast the future. For each change, the simulator can search the proper architecture and the optimal self-management strategies or a set of possible strategies (advices). The ideal MDE toolsets can also be applied to trace SCPS and collect data for self-learning, which forms a close-loop runtime self-evolvable V&V solution.

8. Conclusions and discussion

It is a fine art to develop and maintain a dependable SCPS. In this survey, we first briefly introduced the concepts of CPS, self-management and dependability, discussed the necessary of systematical design of dependable SCPS and the lack of comprehensive survey on the technologies integration of SCPS. We separately investigated and analyzed the current researches on self-adaptive architecture design, self-adaptation strategies, self-healing solutions and the corresponding dependability V&V methods. The dependability of self-management in SCPS depends on the combined actions of architecture (reorganization), the fitness of self-management decision, the timeliness of decision processing, which needs both design period evaluations and runtime assessment. We believe that MDE and model@run.time will be the trend for dependability V&V at the design period and at runtime. Furthermore, we can integrate MDE and model@run.time to build all-in one solution for future SCPS design and management.

The increasing complexity of SCPS and the uncertainty of future environment are the two main error sources of dependability. To tame the complexity, the two principles are 1) using simplicity to control complexity, 2) applying self-management to reduce the complexity of runtime management. Numbers technologies are available to reduce the complexity of design, such as, formal model, standardization, abstraction and decoupling, etc. Moreover, we can also apply software defined architecture (SDA) to archive self-adaptive architecture, decouple the control logic from normal functions, and improve the C&C of subsystems. To achieve best balance, we need to systematically consider the side-effects of every adopted technologies, compromise between the complexity and flexibility/adaptability, and then elaborately and rigorously integrate these technologies.

To reduce the uncertainties, the self-management decisions should be gradual refined with the progress. In detail, the DSS can make prophetic self-management contracts, apply requirements@run.time methods to instruct the cooperation among the local subsystems and let the actuator make the final decision whether to take activates or not. To deal with uncertainties, the local systems have enough autonomy to decide to accept or reject the contracts. They should also be able to refine process flow and related thresholds of the contracts based on their newest situations and observations.

Models play important roles in dependable SCPS design and safe self-management decision making. Considering the synergy, we should co-verify the dependability/safety of self-adaptive

architecture and the self-management strategies, and check the best collaboration for different situations. As a systematical solution, MDE can improve the dependability from architecture and strategies design to the cooperation between teams and the results of validation. Model@run.time is another systematical solution to improve the dependability of self-management decision and the related procedure of decision process. With model@run.time solution, SCPS can model the statues of cyber space and physical space, and then make the best suitable contracts and specify the requirements of contract execution, so that the distributed subsystems can process the contract dependably and timely. However, current MDE tools and model@run.time approaches are fragmented and dedicated to specified scenarios, none of them can meet the requirement of systematical evaluation of SCPS.

Besides the traditional measures, we can also apply newest technologies to improve the dependability. Such as applying design diversity to improve fault tolerance, using virtualization technologies to simplify fault isolation and application scheduling, using middleware technologies to guarantee the reliability and consistency of operations among heterogeneous subsystems, dynamic multi-object scheduling to balance the different requirements (i.e. short-term requirements: timely, performance and dependability, and long-term requirements: energy and survivability). Moreover, we should take full advantages of characteristics of SCPS, such as self-similarity of architecture and behavior and the massive (heterogeneous) redundant subsystems, to simplify the management and improve the dependability of SCPS. E.g. SCPS can use the transfer learning method to learn the model/behavior from similar subsystem, the new joined node can learn from neighbors with the same role and initiate its own parameters. Meanwhile, the subsystem with same role can check the similarity of each other's behavior to detect and prevent failures.

Domain knowledge, rules and expert system are the mainstream solutions to make the self-management decisions in current SCPS, and these solutions also main means to verify the safety and fitness of decisions at runtime. However, these methods are dedicated to predefined known scenarios, which can't handle the unknown situations. To overcome this issue, more and more researchers pay attention to "AI" based or data-driven methods, and explore to use these methods to build a self-leaning system to improve the adaptability and intelligence of self-management decisions making. However, the failure data (error behavior) is relatively scarce and different to be collected, which also challenges the application of "AI" based solutions on safety V&V. Moreover, most of "AI" methods are weak in interpretability, the related (safety) V&V method is rare.

Moreover, as an inference of Turing halting problem, it is impossible to build an absolutely dependable self-healing solution. Consequently, SCPS should estimate the cost of each self-healing decision, and let the subsystem crash safely in some cases. As the final backup healing solution, manual maintenance is still the essential option. Hence, we need to improve the observability and traceability of SCPS. Building such a user-friendly human-in-loop SCPS can also reduce the failure caused by human beings.

For the past decade, SCPS becomes more and more popular. Researchers have explored massive theories and tools to improve the quality of SCPS. However, these researches are fragmented. Considering the complexity of SCPS and MDE toolset, it needs the experts from different disciplines to build the all-in one solution to guarantee the dependability through the whole life-cycle of SCPS. And considering the huge workload of technology integration, interagency and cross-disciplinary cooperation are urgently necessary to promote the researches of dependable SCPS.

Acknowledgments: This work was supported by a grant from National High Technology Research and Development Program of China (863 Program) (No. 2013AA01A205), and National Natural Science Foundation of China (No. 61370085) and Provincial Science & Technology Pillar Program of HUBEI (nos. 2017AAA027, 2017AAA042); Provincial Science & Technology International Cooperation Program of HUBEI: (no. 2017AHB048).

Author Contributions: The survey is a product of the intellectual collaboration of all authors. Z.P. contributed to investigation and writing; Z.D.C. and K.M.H supervised the work; Z.D.C. and K.M.H and Z.Z. contributed to manuscript organization; D.J. and L.J.J and Z.H.Y. provided insightful comments and suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

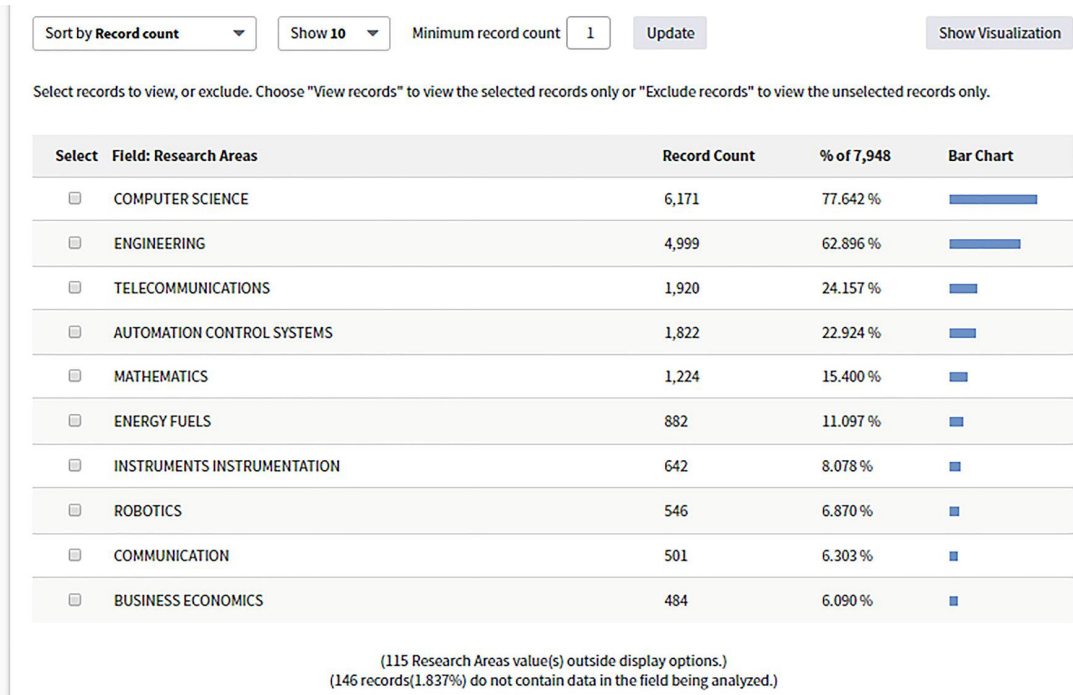


Figure 14. Top 10 research areas of CPS on the web of science, due 2018-8-30

According the statistic result of “web of science”, CPS has been widely applied in 115 fields, which range from “computer science” to “sport science”. The publications in the decade years are shown in Figure 14. I.e. according to “web of science”, there are 7948 papers returned with the keyword “cyber physical systems” or “cyber physical system” due 2018-8-30, and the amount of publication increases rapidly in recent years. The top 10 research areas of CPS are illustrated in

Figure 15. CPSs are integrations of computation, networking, and physical processes, which just corresponds to the top 4 areas, where “computer science” corresponds computation, “engineering” is about integrations, “telecommunications” is for networking, and “automation control systems” focuses on physical processes. As shown in Figure 16, the publication on the dependability of CPS rapidly increased in recent years, and the amount is 456 due 2018-8-30. According to further review, there are 67 papers of them discussing dependability (or related topic) in one or more sections, and 52 papers explicitly including "dependability", "reliability" and "availability" in their keyword fields. As CPS is firstly proposed in 2006, the publication year is from 2006 to 2018.

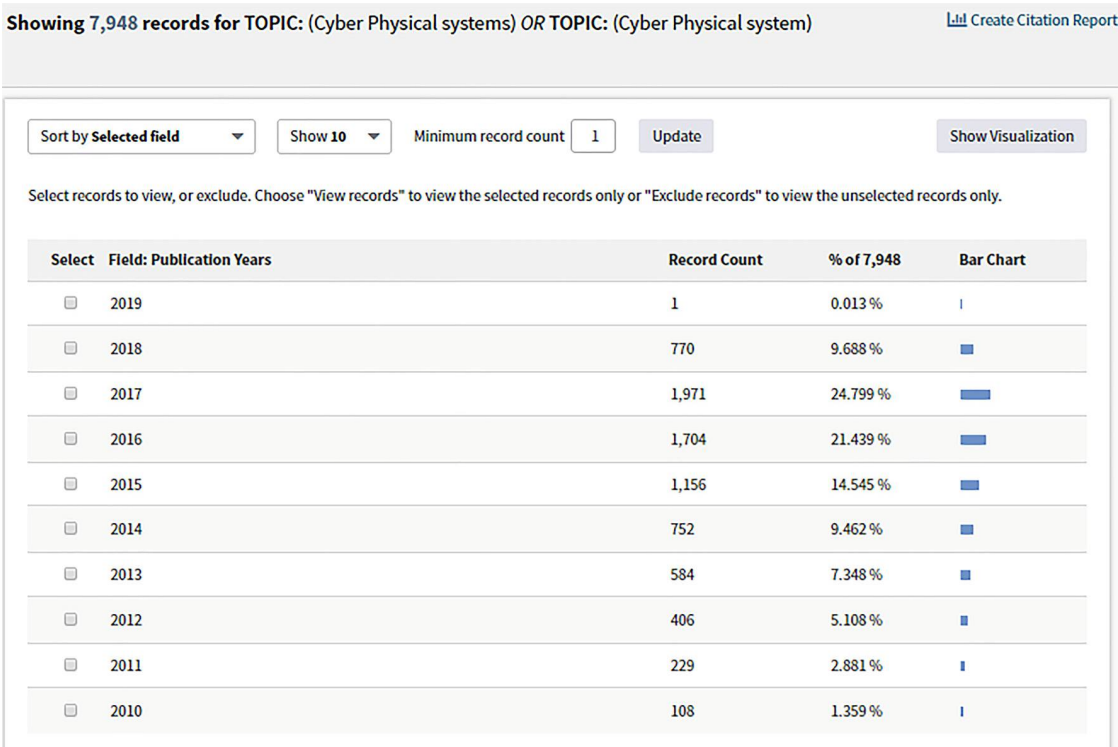


Figure 15 Published papers on CPS by year on the web of science, due 2018-8-30

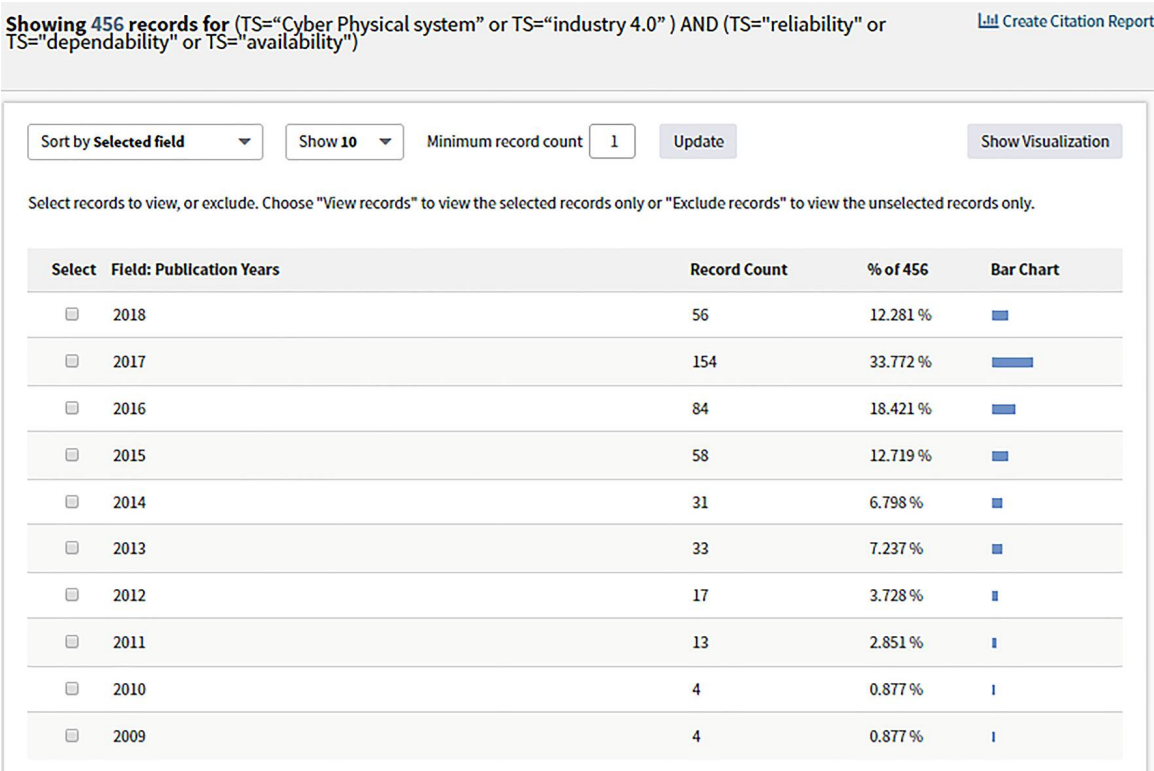


Figure 16 Published papers on dependability and CPS by year on the web of science, due 2018-8-30

1848 **Table 4.** The statistics of publications on dependable CPS (due 2018-8-30)

Keywords	Web of Science	IEEE Xplore	ACM	ScienceDirect
CPS	7, 948 (Topic)	7,902 (meta data only) 2,207 (keyword)	9,470 (full text) 545(keyword)	11,061(all items) 1,227 (keyword)
Dependability &CPS	36 (Topic)	99 (meta data only) 35 (keyword)	5,989 (full text) 32 (keyword)	701(all items) 21(keyword)
Reliability & CPS	343 (Topic)	854 (meta data only) 101 (keyword)	914 (full text) 89 (keyword)	3,806(all items) 61(keyword)
Availability & CPS	109 (Topic)	225 (meta data only) 79 (keyword)	819(full text) 11 (keyword)	8,981(all items) 114(keyword)
Safety & CPS	335 (Topic)	827 (meta data only) 268 (keyword)	699 (full text) 157 (keyword)	4,022(all items) 107(keyword)
Realtime & Adaptation ¹	242 (Topic)	176 (meta data only) 3 (keyword)	350 (full text) 125(keyword)	1,745(all items) 27(keyword)
Rt & adp* & dep* ²	14 (Topic)	0 (meta data only) 0 (keyword)	26(full text) 0 (keyword)	702(all items) 1(keyword)
Rt &CPS & adp* & dep* ³	0 (Topic)	0 0	25(full text) 0	48 (all items) 0

1849 The search filed on “web of science” is topic, which includes title, abstract, author keywords, keywords plus;
1850 “web of science” doesn’t support to search within the field of “Author Keywords”. The “meta data only” field
1851 on “IEEE Xplore Digital Library” includes the abstract, index terms, and bibliographic citation data. The
1852 “full-text” field on “ACM Digital Library” includes the abstract, index terms, text. The “all items” field on
1853 ScienceDirect includes all parts of the document. The “Author Keywords” field means the keyword that
1854 author listed in paper. To ScienceDirect, the field of “Keyword” includes Title, abstract, and keywords.

1855 ¹ The keywords are “realtime AND (adaptation or self-adaptation OR self-adaptive)”

1856 ² The keywords of (Rt & adp* & dep*) are “(realtime) AND (adaptation OR self-adaptation OR self-adaptive)
1857 AND (safety OR dependability OR reliability)”

1858 ³ The keywords of (Rt & CPS & adp* & dep*) are “(realtime) AND (cyber physical system) AND (adaptation OR
1859 self-adaptation OR self-adaptive) AND (safety OR dependability OR reliability)”

1860

References

1. Mosterman, P. J.; Zander, J., Industry 4.0 as a cyber-physical system study. *Software & Systems Modeling* **2016**, *15* (1), 17-29.
2. Derler, P.; Lee, E. A.; Vincentelli, A. S., Modeling cyber-physical systems. *Proceedings of the IEEE* **2012**, *100* (1), 13-28.
3. Webinar, K. NIST Cyber-Physical Systems Public Working Group (CPS PWG). <https://www.nist.gov/sites/default/files/documents/el/CPS-PWG-Kickoff-Webinar-Presentation-FINAL.PDF>.
4. Rajkumar, R. R.; Lee, I.; Sha, L.; Stankovic, J. In *Cyber-physical systems: the next computing revolution*, Proceedings of the 47th Design Automation Conference, ACM: 2010; pp 731-736.
5. Gunes, V.; Peter, S.; Givargis, T.; Vahid, F., A Survey on Concepts, Applications, and Challenges in Cyber-Physical Systems. *KSII Transactions on Internet and Information Systems (TIIS)* **2014**, *8* (12), 4242-4268.
6. Leitão, P.; Colombo, A. W.; Karnouskos, S., Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry* **2016**, *81*, 11-25.
7. de Brito, M. S.; Hoque, S.; Steinke, R.; Willner, A., Towards Programmable Fog Nodes in Smart Factories. *2016 Ieee 1st International Workshops on Foundations and Applications of Self* Systems (Fas*W)* **2016**, 236-241.
8. Wan, J.; Chen, M.; Xia, F.; Di, L.; Zhou, K., From machine-to-machine communications towards cyber-physical systems. *Computer Science and Information Systems* **2013**, *10* (3), 1105-1128.
9. Ceccarelli, A.; Bondavalli, A.; Froemel, B.; Hoeffberger, O.; Kopetz, H., Basic Concepts on Systems of Systems. In *Cyber-Physical Systems of Systems: Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy*, Bondavalli, A.; Bouchenak, S.; Kopetz, H., Eds. Springer International Publishing: Cham, 2016; pp 1-39.
10. Lu, C.; Saifullah, A.; Li, B.; Sha, M.; Gonzalez, H.; Gunatilaka, D.; Wu, C.; Nie, L.; Chen, Y., Real-time wireless sensor-actuator networks for industrial cyber-physical systems. *Proceedings of the IEEE* **2016**, *104* (5), 1013-1024.
11. Zhang, X.-M.; Han, Q.-L.; Yu, X., Survey on Recent Advances in Networked Control Systems. *Ieee Transactions on Industrial Informatics* **2016**, *12* (5), 1740-1752.
12. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M., Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems* **2013**, *29* (7), 1645-1660.
13. Giordano, A.; Spezzano, G.; Vinci, A. In *Smart Agents and Fog Computing for Smart City Applications*, International Conference on Smart Cities, Springer: 2016; pp 137-146.
14. Zhang, Y.; Qian, C.; Lv, J.; Liu, Y., Agent and cyber-physical system based self-organizing and self-adaptive intelligent shopfloor. *IEEE Transactions on Industrial Informatics* **2016**.

15. Eliasson, J.; Delsing, J.; Derhamy, H.; Salcic, Z.; Wang, K. In *Towards industrial Internet of Things: An efficient and interoperable communication framework*, Industrial Technology (ICIT), 2015 IEEE International Conference on, IEEE: 2015; pp 2198-2204.
16. Agirre, A.; Parra, J.; Armentia, A.; Estévez, E.; Marcos, M., QoS Aware Middleware Support for Dynamically Reconfigurable Component Based IoT Applications. *International Journal of Distributed Sensor Networks* **2016**.
17. Salehie, M.; Tahvildari, L., Self-Adaptive Software: Landscape and Research Challenges. *Acm Transactions on Autonomous and Adaptive Systems* **2009**, 4 (2).
18. Kephart, J. O.; acm, Research challenges of autonomic computing. In *Icse 05: 27th International Conference on Software Engineering, Proceedings*, 2005; pp 15-22.
19. Krupitzer, C.; Roth, F. M.; VanSyckel, S.; Schiele, G.; Becker, C., A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* **2015**, 17, 184-206.
20. Alegre, U.; Augusto, J. C.; Clark, T., Engineering context-aware systems and applications: A survey. *Journal of Systems and Software* **2016**, 117, 55-83.
21. Kang, H. S.; Lee, J. Y.; Choi, S.; Kim, H.; Park, J. H.; Son, J. Y.; Kim, B. H.; Do Noh, S., Smart manufacturing: Past research, present findings, and future directions. *International Journal of Precision Engineering and Manufacturing-Green Technology* **2016**, 3 (1), 111-128.
22. Sanislav, T.; Miclea, L., Cyber-physical systems-concept, challenges and research areas. *Journal of Control Engineering and Applied Informatics* **2012**, 14 (2), 28-33.
23. Broy, M.; Cengarle, M. V.; Geisberger, E. In *Cyber-physical systems: imminent challenges*, Monterey workshop, Springer: 2012; pp 1-28.
24. Crnkovic, I.; Malavolta, I.; Muccini, H.; Sharaf, M.; Ieee, On the Use of Component-Based Principles and Practices for Architecting Cyber-Physical Systems. *Proceedings 2016 19th International Acm Sigsoft Symposium on Component-Based Software Engineering* **2016**, 23-32.
25. Völpl, M.; Asmussen, N.; Härtig, H.; Nöthen, B.; Fettweis, G. In *Towards dependable CPS infrastructures: Architectural and operating-system challenges*, Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, IEEE: 2015; pp 1-8.
26. Avižienis, A.; Laprie, J.-C.; Randell, B., Dependability and its threats: a taxonomy. In *Building the Information Society*, Springer: 2004; pp 91-120.
27. Warriach, E. U.; Ozcelebi, T.; Lukkien, J. J. In *Self-* Properties in Smart Environments: Requirements and Performance Metrics*, Intelligent Environments (Workshops), 2014; pp 194-205.
28. Psailer, H.; Dustdar, S., A survey on self-healing systems: approaches and systems. *Computing* **2011**, 91 (1), 43-73.
29. Sha, L., Using simplicity to control complexity. *IEEE Software* **2001**, (4), 20-28.
30. Benosman, M., A survey of some recent results on nonlinear fault tolerant control. *Mathematical Problems in Engineering* **2009**, 2010.

- 1942 31. Turing, A. M., On computable numbers, with an application to the
1943 Entscheidungsproblem. *Proceedings of the London mathematical society* **1937**, 2 (1),
1944 230-265.
- 1945 32. Ohmann, P.; Brown, D. B.; Neelakandan, N.; Linderth, J.; Liblit, B., Optimizing
1946 Customized Program Coverage. In *2016 31st Ieee/Acm International Conference on*
1947 *Automated Software Engineering*, Lo, D.; Apel, S.; Khurshid, S., Eds. 2016; pp 27-38.
- 1948 33. Tsigkanos, C.; Kehrer, T.; Ghezzi, C., Architecting dynamic cyber-physical spaces.
1949 *Computing* **2016**, 98 (10), 1011-1040.
- 1950 34. Broman, D.; Derler, P.; Eidson, J., Temporal issues in cyber-physical systems. *Journal*
1951 *of the Indian Institute of Science* **2013**, 93 (3), 389-402.
- 1952 35. Novoa-Hernández, P.; Corona, C. C.; Pelta, D. A., Self-adaptation in dynamic
1953 environments-a survey and open issues. *International Journal of Bio-Inspired Computation*
1954 **2016**, 8 (1), 1-13.
- 1955 36. Alho, P.; Mattila, J., Service-oriented approach to fault tolerance in CPSs. *Journal of*
1956 *Systems and Software* **2015**, 105, 1-17.
- 1957 37. Pfrommer, J.; Stogl, D.; Aleksandrov, K.; Navarro, S. E.; Hein, B.; Beyerer, J., Plug &
1958 produce by modelling skills and service-oriented orchestration of reconfigurable
1959 manufacturing systems. *At-Automatisierungstechnik* **2015**, 63 (10), 790-800.
- 1960 38. Dai, W.; Huang, W.; Vyatkin, V.; Ieee, Knowledge-Driven Service Orchestration
1961 Engine for Flexible Information Acquisition in Industrial Cyber-Physical Systems. In
1962 *Proceedings 2016 Ieee 25th International Symposium on Industrial Electronics*, 2016; pp
1963 1055-1060.
- 1964 39. Leitao, P.; Karnouskos, S.; Ribeiro, L.; Lee, J.; Strasser, T.; Colombo, A. W., Smart
1965 Agents in Industrial Cyber-Physical Systems. *Proceedings of the IEEE* **2016**, 104 (5),
1966 1086-1101.
- 1967 40. Lee, J.; Ardakani, H. D.; Yang, S.; Bagheri, B., Industrial big data analytics and
1968 cyber-physical systems for future maintenance & service innovation. *Procedia CIRP* **2015**,
1969 38, 3-7.
- 1970 41. Wang, L.; Haghighi, A., Combined strength of holons, agents and function blocks in
1971 cyber-physical systems. *Journal of Manufacturing Systems* **2016**, 40, 25-34.
- 1972 42. Pautasso, C.; Zimmermann, O.; Leymann, F. In *Restful web services vs. big'web*
1973 *services: making the right architectural decision*, Proceedings of the 17th international
1974 conference on World Wide Web, ACM: 2008; pp 805-814.
- 1975 43. Qin, J.; Ma, Q.; Shi, Y.; Wang, L., Recent Advances in Consensus of Multi-Agent
1976 Systems: A Brief Survey. *Ieee Transactions on Industrial Electronics* **2017**, 64 (6),
1977 4972-4983.
- 1978 44. Ye, L.; Qian, K.; Zhang, L., S-PDH: A CPS Service Contract Framework for
1979 Composition. In *Service-Oriented Computing - Icsoc 2015 Workshops*, Norta, A.; Gaaloul,
1980 W.; Gangadharan, G. R.; Dam, H. K., Eds. 2016; Vol. 9586, pp 79-90.
- 1981 45. Brizzi, P.; Conzon, D.; Khaleel, H.; Tomasi, R.; Pastrone, C.; Spirito, A.; Knechtel, M.;
1982 Pramudianto, F.; Cultrona, P. In *Bringing the Internet of Things along the manufacturing*
1983 *line: A case study in controlling industrial robot and monitoring energy consumption*

- 1984 *remotely*, Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th
1985 Conference on, IEEE: 2013; pp 1-8.
- 1986 46. Wang, S.; Wan, J.; Zhang, D.; Li, D.; Zhang, C., Towards smart factory for Industry
1987 4.0: A self-organized multi-agent system with big data based feedback and coordination.
1988 *Computer Networks* **2016**, *101*, 158-168.
- 1989 47. Shu, Z.; Wan, J.; Zhang, D.; Li, D., Cloud-Integrated Cyber-Physical Systems for
1990 Complex Industrial Applications. *Mobile Networks & Applications* **2016**, *21* (5), 865-878.
- 1991 48. Perez, H.; Javier Gutierrez, J.; Peiro, S.; Crespo, A., Distributed architecture for
1992 developing mixed-criticality systems in multi-core platforms. *Journal of Systems and*
1993 *Software* **2017**, *123*, 145-159.
- 1994 49. Jablkowski, B.; Gabor, U. T.; Spinczyk, O., Evolutionary planning of virtualized
1995 cyber-physical compute and control clusters. *Journal of Systems Architecture* **2017**, *73*,
1996 17-27.
- 1997 50. Wan, J.; Tang, S.; Shu, Z.; Li, D.; Wang, S.; Imran, M.; Vasilakos, A. V.,
1998 Software-Defined Industrial Internet of Things in the Context of Industry 4.0. *Ieee Sensors*
1999 *Journal* **2016**, *16* (20), 7373-7380.
- 2000 51. Bruton, K.; Walsh, B. P.; Cusack, D. O.; O'Donovan, P.; O'Sullivan, D. T. J., Enabling
2001 Effective Operational Decision Making on a Combined Heat and Power System using the
2002 5C Architecture. In *5th Cirp Global Web Conference - Research and Innovation for Future*
2003 *Production*, Alexopoulos, K., Ed. 2016; Vol. 55, pp 296-301.
- 2004 52. Camara, J.; Correia, P.; de Lemos, R.; Garlan, D.; Gomes, P.; Schmerl, B.; Ventura, R.,
2005 Incorporating architecture-based self-adaptation into an adaptive industrial software system.
2006 *Journal of Systems and Software* **2016**, *122*, 507-523.
- 2007 53. Cámara, J.; de Lemos, R.; Laranjeiro, N.; Ventura, R.; Vieira, M. In *Robustness*
2008 *evaluation of the rainbow framework for self-adaptation*, Proceedings of the 29th Annual
2009 ACM Symposium on Applied Computing, ACM: 2014; pp 376-383.
- 2010 54. Kit, M.; Gerostathopoulos, I.; Bures, T.; Hnetyinka, P.; Plasil, F. In *An architecture*
2011 *framework for experimentations with self-adaptive cyber-physical systems*, 2015
2012 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and
2013 Self-Managing Systems, IEEE: 2015; pp 93-96.
- 2014 55. Bures, T.; Hnetyinka, P.; Plasil, F. In *Strengthening architectures of smart CPS by*
2015 *modeling them as runtime product-lines*, Proceedings of the 17th international ACM
2016 Sigsoft symposium on Component-based software engineering, ACM: 2014; pp 91-96.
- 2017 56. Masrur, A.; Kit, M.; Matěna, V.; Bureš, T.; Hardt, W., Component-based design of
2018 cyber-physical applications with safety-critical requirements. *Microprocessors and*
2019 *Microsystems* **2016**, *42*, 70-86.
- 2020 57. Gerostathopoulos, I.; Bures, T.; Hnetyinka, P.; Keznikl, J.; Kit, M.; Plasil, F.; Plouzeau,
2021 N., Self-adaptation in software-intensive cyber-physical systems: From system goals to
2022 architecture configurations. *Journal of Systems and Software* **2016**, *122*, 378-397.
- 2023 58. Iarovyi, S.; Mohammed, W. M.; Lobov, A.; Ferrer, B. R.; Lastra, J. L. M.,
2024 Cyber-Physical Systems for Open-Knowledge-Driven Manufacturing Execution Systems.
2025 *Proceedings of the Ieee* **2016**, *104* (5), 1142-1154.

- 2026 59. Park, J.; Lee, S.; Yoon, T.; Kim, J. M., An autonomic control system for high-reliable
2027 CPS. *Cluster Computing* **2015**, *18* (2), 587-598.
- 2028 60. Axelsson, J.; Kobetski, A. In *Architectural Concepts for Federated Embedded Systems*,
2029 Proceedings of the 2014 European Conference on Software Architecture Workshops, ACM:
2030 2014; p 25.
- 2031 61. Gang, L.; GuPing, Z. In *Self-Reconfiguration Architecture for Distribution Automation*
2032 *System Based on Cyber-Physical*, Power and Energy Engineering Conference (APPEEC),
2033 2012 Asia-Pacific, IEEE: 2012; pp 1-4.
- 2034 62. Pajic, M.; Chernoguzov, A.; Mangharam, R., Robust Architectures for Embedded
2035 Wireless Network Control and Actuation. *Acm Transactions on Embedded Computing*
2036 *Systems* **2012**, *11* (4).
- 2037 63. Lee, E. A.; Seshia, S. A., *Introduction to embedded systems: A cyber-physical systems*
2038 *approach*. MIT Press: 2016.
- 2039 64. Qadir, J.; Ahmed, N.; Ahad, N., Building programmable wireless networks: an
2040 architectural survey. *EURASIP Journal on Wireless Communications and Networking* **2014**,
2041 *2014* (1), 172.
- 2042 65. Saha, I.; Roy, S.; Ramesh, S., Formal Verification of Fault-Tolerant Startup
2043 Algorithms for Time-Triggered Architectures: A Survey. *Proceedings of the IEEE* **2016**,
2044 *104* (5), 904-922.
- 2045 66. Kharchenko, V. In *Diversity for safety and security of embedded and cyber physical*
2046 *systems: Fundamentals review and industrial cases*, Electronics Conference (BEC), 2016
2047 15th Biennial Baltic, IEEE: 2016; pp 17-26.
- 2048 67. Sterbenz, J. P. G.; Hutchison, D.; Cetinkaya, E. K.; Jabbar, A.; Rohrer, J. P.; Schoeller,
2049 M.; Smith, P., Redundancy, diversity, and connectivity to achieve multilevel network
2050 resilience, survivability, and disruption tolerance invited paper. *Telecommunication*
2051 *Systems* **2014**, *56* (1), 17-31.
- 2052 68. Rodriguez, M.; Kwiat, K. A.; Kamhoua, C. A. In *On the use of design diversity in fault*
2053 *tolerant and secure systems: A qualitative analysis*, Computational Intelligence for Security
2054 and Defense Applications (CISDA), 2015 IEEE Symposium on, IEEE: 2015; pp 1-8.
- 2055 69. Rodriguez, M.; Kwiat, K. A.; Kamhoua, C. A. In *Modeling fault tolerant architectures*
2056 *with design diversity for secure systems*, Military Communications Conference, MILCOM
2057 2015-2015 IEEE, IEEE: 2015; pp 1254-1263.
- 2058 70. Lien, S.-Y., Resource-optimal heterogeneous machine-to-machine communications in
2059 software defined networking cyber-physical systems. *Wireless Personal Communications*
2060 **2015**, *84* (3), 2215-2239.
- 2061 71. Iftikhar, M. U.; Weyns, D. In *Activforms: Active formal models for self-adaptation*,
2062 Proceedings of the 9th International Symposium on Software Engineering for Adaptive and
2063 Self-Managing Systems, ACM: 2014; pp 125-134.
- 2064 72. Uddin, I.; Rakib, A.; Ul Haque, H. M., A Framework for Implementing Formally
2065 Verified Resource-Bounded Smart Space Systems. *Mobile Networks & Applications* **2017**,
2066 *22* (2), 289-304.

73. Zhou, P.; Zuo, D.; Hou, K.-M.; Zhang, Z., A Decentralized Compositional Framework for Dependable Decision Process in Self-Managed Cyber Physical Systems. *Sensors (Basel, Switzerland)* **2017**, *17* (11).
74. Musil, A.; Musil, J.; Weyns, D.; Bures, T.; Muccini, H.; Sharaf, M., Patterns for Self-Adaptation in Cyber-Physical Systems. In *Multi-Disciplinary Engineering for Cyber-Physical Production Systems*, Springer: 2017; pp 331-368.
75. Conti, S.; La Corte, A.; Nicolosi, R.; Rizzo, S., Impact of cyber-physical system vulnerability, telecontrol system availability and islanding on distribution network reliability. *Sustainable Energy, Grids and Networks* **2016**, *6*, 143-151.
76. Broman, D.; Zimmer, M.; Kim, Y.; Kim, H.; Cai, J.; Shrivastava, A.; Edwards, S. A.; Lee, E. A. In *Precision timed infrastructure: Design challenges*, Electronic System Level Synthesis Conference (ESLsyn), 2013, IEEE: 2013; pp 1-6.
77. Lee, E. A., Computing needs time. *Communications of the ACM* **2009**, *52* (5), 70-79.
78. Patel, H. D.; Lickly, B.; Burgers, B.; Lee, E. A. *A timing requirements-aware scratchpad memory allocation scheme for a precision timed architecture*; DTIC Document: 2008.
79. Zou, J.; Matic, S.; Lee, E. A. In *PtityOS: A lightweight microkernel for PtiDES real-time systems*, Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th, IEEE: 2012; pp 209-218.
80. Lickly, B.; Liu, I.; Kim, S.; Patel, H. D.; Edwards, S. A.; Lee, E. A. In *Predictable programming on a precision timed architecture*, Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems, ACM: 2008; pp 137-146.
81. Bui, D.; Lee, E.; Liu, I.; Patel, H.; Reineke, J. In *Temporal isolation on multiprocessing architectures*, Proceedings of the 48th Design Automation Conference, ACM: 2011; pp 274-279.
82. Dierikx, E. F.; Wallin, A. E.; Fordell, T.; Myrsky, J.; Koponen, P.; Merimaa, M.; Pinkert, T. J.; Koelemeij, J. C.; Peek, H. Z.; Smets, R., White Rabbit Precision Time Protocol on Long-Distance Fiber Links. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control* **2016**, *63* (7), 945-952.
83. Elattar, M.; Dürkop, L.; Jasperneite, J. In *Utilizing LTE QoS features to provide a reliable access network for cyber-physical systems*, Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on, IEEE: 2015; pp 956-961.
84. Aristova, N. I., Ethernet in industrial automation: Overcoming obstacles. *Automation and Remote Control* **2016**, *77* (5), 881-894.
85. Watteyne, T.; Handziski, V.; Vilajosana, X.; Duquennoy, S.; Hahm, O.; Baccelli, E.; Wolisz, A., Industrial wireless ip-based cyber-physical systems. *Proceedings of the IEEE* **2016**, *104* (5), 1025-1038.
86. Eidson, J.; Lee, E. A.; Matic, S.; Seshia, S. A.; Zou, J. In *A time-centric model for cyber-physical applications*, Workshop on Model Based Architecting and Construction of Embedded Systems (ACES-MB), 2010; pp 21-35.

- 2108 87. Abella, J.; Hernandez, C.; Quiñones, E.; Cazorla, F. J.; Conmy, P. R.;
2109 Azkarate-askasua, M.; Perez, J.; Mezzetti, E.; Vardanega, T. In *WCET analysis methods:
2110 Pitfalls and challenges on their trustworthiness*, 10th IEEE International Symposium on
2111 Industrial Embedded Systems (SIES), IEEE: 2015; pp 1-10.
- 2112 88. Wilhelm, R.; Engblom, J.; Ermedahl, A.; Holsti, N.; Thesing, S.; Whalley, D.; Bernat,
2113 G.; Ferdinand, C.; Heckmann, R.; Mitra, T., The worst-case execution-time
2114 problem—overview of methods and survey of tools. *ACM Transactions on Embedded
2115 Computing Systems (TECS)* **2008**, *7* (3), 36.
- 2116 89. Krishna, C., Fault-tolerant scheduling in homogeneous real-time systems. *ACM
2117 Computing Surveys (CSUR)* **2014**, *46* (4), 48.
- 2118 90. Nelaturi, S.; de Kleer, J.; Shapiro, V.; Ieee, Combinatorial Models for Heterogeneous
2119 System Composition and Analysis. *2016 11th Systems of System Engineering Conference
2120 (Sose), Ieee* **2016**.
- 2121 91. Gössler, G.; Sifakis, J., Composition for component-based modeling. *Science of
2122 Computer Programming* **2005**, *55* (1-3), 161-183.
- 2123 92. Sztipanovits, J.; Koutsoukos, X.; Karsai, G.; Kottenstette, N.; Antsaklis, P.; Gupta, V.;
2124 Goodwine, B.; Baras, J.; Wang, S., Toward a science of cyber–physical system integration.
2125 *Proceedings of the IEEE* **2011**, *100* (1), 29-44.
- 2126 93. Nuzzo, P.; Sangiovanni-Vincentelli, A. L.; Bresolin, D.; Geretti, L.; Villa, T., A
2127 platform-based design methodology with contracts and related tools for the design of
2128 cyber-physical systems. *Proceedings of the IEEE* **2015**, *103* (11), 2104-2132.
- 2129 94. Attie, P.; Baranov, E.; Bliudze, S.; Jaber, M.; Sifakis, J., A general framework for
2130 architecture composability. *Formal Aspects of Computing* **2016**, *28* (2), 207-231.
- 2131 95. Seshia, S. A., Combining induction, deduction, and structure for verification and
2132 synthesis. *Proceedings of the IEEE* **2015**, *103* (11), 2036-2051.
- 2133 96. Tripakis, S., Compositionality in the science of system design. *Proceedings of the
2134 IEEE* **2016**, *104* (5), 960-972.
- 2135 97. Chen, T.; Chilton, C.; Jonsson, B.; Kwiatkowska, M. In *A compositional specification
2136 theory for component behaviours*, European Symposium on Programming, Springer: 2012;
2137 pp 148-168.
- 2138 98. Le, T. T. H.; Passerone, R.; Fahrenberg, U.; Legay, A., Contract-Based Requirement
2139 Modularization via Synthesis of Correct Decompositions. *Acm Transactions on Embedded
2140 Computing Systems* **2016**, *15* (2).
- 2141 99. Gerostathopoulos, I.; Bures, T.; Hnetynka, P.; Keznikl, J.; Kit, M.; Plasil, F.; Plouzeau,
2142 N., Self-adaptation in software-intensive cyber–physical systems: From system goals to
2143 architecture configurations. *Journal of Systems and Software* **2016**.
- 2144 100. Laird, L. M.; Brennan, M. C., *Software measurement and estimation: a practical
2145 approach*. John Wiley & Sons: 2006; Vol. 2.
- 2146 101. Henry, S.; Kafura, D., Software structure metrics based on information flow. *IEEE
2147 transactions on Software Engineering* **1981**, (5), 510-518.
- 2148 102. Halstead, M. H., *Elements of Software Science (Operating and programming
2149 systems series)*. Elsevier Science Inc., New York, NY: 1977.

- 2150 103. Isermann, R., *Fault-diagnosis systems: an introduction from fault detection to fault*
2151 *tolerance*. Springer Science & Business Media: 2006.
- 2152 104. Stapelberg, R. F., *Handbook of reliability, availability, maintainability and safety*
2153 *in engineering design*. Springer Science & Business Media: 2009.
- 2154 105. Birolini, A., *Reliability engineering: theory and practice*. 7th(2014) ed.; Springer,
2155 New York, NY: 2002.
- 2156 106. Hasan, O.; Ahmed, W.; Tahar, S.; Hamdi, M. S.; Simos, T. E.; Tsitouras, C. In
2157 *Reliability block diagrams based analysis: A survey*, AIP Conference Proceedings, AIP
2158 Publishing: 2015; p 850129.
- 2159 107. Ruijters, E.; Stoelinga, M., *Fault tree analysis: A survey of the state-of-the-art in*
2160 *modeling, analysis and tools*. *Computer Science Review* **2015**, 15-16, 29-62.
- 2161 108. Kabir, S., *An overview of fault tree analysis and its application in model based*
2162 *dependability analysis*. *Expert Systems with Applications* **2017**, 77, 114-135.
- 2163 109. Höfig, K.; Domis, D. In *Failure-dependent execution time analysis*, Proceedings of
2164 the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on
2165 Quality of software architectures--QoSA and architecting critical systems--ISARCS, ACM:
2166 2011; pp 115-122.
- 2167 110. Ahmed, W.; Hasan, O.; Tahar, S. In *Formal dependability modeling and analysis:*
2168 *a survey*, International Conference on Intelligent Computer Mathematics, Springer: 2016;
2169 pp 132-147.
- 2170 111. Ma, Z.; Fu, X.; Yu, Z. In *Object-oriented petri nets based formal modeling for*
2171 *high-confidence cyber-physical systems*, Wireless Communications, Networking and
2172 Mobile Computing (WiCOM), 2012 8th International Conference on, IEEE: 2012; pp 1-4.
- 2173 112. Bryant, B. R.; Gray, J.; Mernik, M.; Clarke, P. J.; France, R. B.; Karsai, G.,
2174 *Challenges and Directions in Formalizing the Semantics of Modeling Languages*.
2175 *Computer Science and Information Systems* **2011**, 8 (2), 225-253.
- 2176 113. Abbas, A. M.; Tsang, E. P.; Nasri, A. H., *Depict: A high-level formal language for*
2177 *modeling constraint satisfaction problems*. *International Journal of Automation and*
2178 *Computing* **2008**, 5 (2), 208-216.
- 2179 114. Feiler, P. H.; Lewis, B. A.; Vestal, S. In *The SAE Architecture Analysis & Design*
2180 *Language (AADL) a standard for engineering performance critical systems*, Computer
2181 Aided Control System Design, 2006 IEEE International Conference on Control
2182 Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE,
2183 IEEE: 2006; pp 1206-1211.
- 2184 115. Aissa, A. B.; Abercrombie, R. K.; Sheldon, F. T.; Mili, A. In *Quantifying the*
2185 *impact of unavailability in cyber-physical environments*, Computational Intelligence in
2186 Cyber Security (CICS), 2014 IEEE Symposium on, IEEE: 2014; pp 1-8.
- 2187 116. Kozat, U. C.; Liang, G.; Kokten, K.; Tapolcai, J., *On Optimal Topology*
2188 *Verification and Failure Localization for Software Defined Networks*. *Ieee-Acm*
2189 *Transactions on Networking* **2016**, 24 (5), 2931-2944.

117. Rungger, M.; Tabuada, P. In *A symbolic approach to the design of robust cyber-physical systems*, Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on, IEEE: 2013; pp 3932-3937.
118. Takai, S.; Kumar, R. In *Verification of generalized inference diagnosability for decentralized diagnosis in discrete event systems*, Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on, IEEE: 2015; pp 1-8.
119. Ganin, A. A.; Massaro, E.; Gutfraind, A.; Steen, N.; Keisler, J. M.; Kott, A.; Mangoubi, R.; Linkov, I., Operational resilience: concepts, design and analysis. *Scientific reports* **2016**, 6.
120. Laibinis, L.; Klionskiy, D.; Troubitsyna, E.; Dorokhov, A.; Lilius, J.; Kupriyanov, M., Modelling Resilience of Data Processing Capabilities of CPS. In *Software Engineering for Resilient Systems*, Majzik, I.; Vieira, M., Eds. 2014; Vol. 8785, pp 55-70.
121. Mashkoor, A.; Hasan, O. In *Formal probabilistic analysis of cyber-physical transportation systems*, International Conference on Computational Science and Its Applications, Springer: 2012; pp 419-434.
122. Marashi, K.; Sarvestani, S. S. In *Towards comprehensive modeling of reliability for smart grids: Requirements and challenges*, High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on, IEEE: 2014; pp 105-112.
123. Han, Y.; Wen, Y.; Guo, C.; Huang, H., Incorporating Cyber Layer Failures in Composite Power System Reliability Evaluations. *Energies* **2015**, 8 (9), 9064-9086.
124. Hu, X.; Liu, S.; Chen, G.; Jiang, C. In *Dependability Modelling and Evaluation of Cyber-Physical Systems: A Model-Driven Perspective*, 1st International Workshop on Cloud Computing and Information Security, Atlantis Press: 2013.
125. Di Giandomenico, F.; Kwiatkowska, M.; Martinucci, M.; Masci, P.; Qu, H., Dependability Analysis and Verification for CONNECTed Systems. In *Leveraging Applications of Formal Methods, Verification, and Validation, Pt Ii*, Margaria, T.; Steffen, B., Eds. 2010; Vol. 6416, pp 263-+.
126. Sanislav, T.; Mois, G.; Miclea, L., An approach to model dependability of cyber-physical systems. *Microprocessors and Microsystems* **2016**, 41, 67-76.
127. Bozzano, M.; Cimatti, A.; Katoen, J.-P.; Nguyen, V. Y.; Noll, T.; Roveri, M., Safety, dependability and performance analysis of extended AADL models. *The Computer Journal* **2011**, 54 (5), 754-775.
128. Esteve, M.-A.; Katoen, J.-P.; Viet Yen, N.; Postma, B.; Yushtein, Y., Formal Correctness, Safety, Dependability, and Performance Analysis of a Satellite. In *2012 34th International Conference on Software Engineering*, Glinz, M.; Murphy, G.; Pezze, M., Eds. 2012; pp 1022-1031.
129. Dal Lago, L.; Ferrante, O.; Passerone, R.; Ferrari, A., Dependability Assessment of SOA-based CPS with Contracts and Model-Based Fault Injection. *IEEE Transactions on Industrial Informatics* **2017**.
130. Ghezzi, C., Dependability of Adaptable and Evolvable Distributed Systems. In *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems*, Springer: 2016; pp 36-60.

- 2232 131. Tundis, A.; Buffoni, L.; Fritzson, P.; Garro, A., Model-Based Dependability
2233 Analysis of Physical Systems with Modelica. *Modelling and Simulation in Engineering*
2234 **2017**, 2017.
- 2235 132. Provan, G., A Contracts-Based Framework for Systems Modeling and Embedded
2236 Diagnostics. In *Software Engineering and Formal Methods, Sefm 2014*, Canal, C.; Idani, A.,
2237 Eds. 2015; Vol. 8938, pp 131-143.
- 2238 133. Hu, F.; Lu, Y.; Vasilakos, A. V.; Hao, Q.; Ma, R.; Patil, Y.; Zhang, T.; Lu, J.; Li,
2239 X.; Xiong, N. N., Robust cyber-physical systems: Concept, models, and implementation.
2240 *Future Generation Computer Systems* **2016**, 56, 449-475.
- 2241 134. Silva, L. C.; Almeida, H. O.; Perkusich, A.; Perkusich, M., A Model-Based
2242 Approach to Support Validation of Medical Cyber-Physical Systems. *Sensors* **2015**, 15 (11),
2243 27625-27670.
- 2244 135. Sanislav, T.; Mois, G. In *A dependability analysis model in the context of*
2245 *cyber-physical systems*, Carpathian Control Conference (ICCC), 2017 18th International,
2246 IEEE: 2017; pp 146-150.
- 2247 136. Tundis, A.; Ferretto, D.; Garro, A.; Brusa, E.; Mühlhäuser, M. In *Dependability*
2248 *assessment of a deicing system through the RAMSAS method*, Systems Engineering
2249 Symposium (ISSE), 2017 IEEE International, IEEE: 2017; pp 1-8.
- 2250 137. Jiang, Y.; Song, H.; Yang, Y.; Liu, H.; Gu, M.; Guan, Y.; Sun, J.; Sha, L.,
2251 Dependable Model-driven Development of CPS: From Stateflow Simulation to Verified
2252 Implementation. *ACM Trans. Cyber-Phys. Syst.* **2018**, 3 (1), 1-31.
- 2253 138. Tsigkanos, C.; Kehrner, T.; Ghezzi, C., Modeling and verification of evolving
2254 cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of*
2255 *Software Engineering*, ACM: Paderborn, Germany, 2017; pp 38-48.
- 2256 139. Fritzson, P., Model-based development of sustainable cyber-physical systems
2257 including requirement formalization using the openModelica model-based development
2258 toolkit. In *Proceedings of the 10th European Conference on Software Architecture*
2259 *Workshops*, ACM: Copenhagen, Denmark, 2016; pp 1-1.
- 2260 140. Hachicha, M.; Dammak, E.; Halima, R. B.; Kacem, A. H. In *A correct by*
2261 *construction approach for modeling and formalizing self-adaptive systems*, Software
2262 Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing
2263 (SNPD), 2016 17th IEEE/ACIS International Conference on, IEEE: 2016; pp 379-384.
- 2264 141. Arcaini, P.; Riccobene, E.; Scandurra, P., Modeling and validating self-adaptive
2265 service-oriented applications. *ACM SIGAPP Applied Computing Review* **2015**, 15 (3),
2266 35-48.
- 2267 142. Brito, A. V.; Negreiros, A. V.; Roth, C.; Sander, O.; Becker, J.; Ieee, Development
2268 and Evaluation of Distributed Simulation of Embedded Systems using Ptolemy and HLA.
2269 In *17th Ieee/Acm International Symposium on Distributed Simulation and Real Time*
2270 *Applications*, 2013; pp 189-196.
- 2271 143. Conzon, D.; Brizzi, P.; Kasinathan, P.; Pastrone, C.; Pramudianto, F.; Cultrona, P.;
2272 Ieee, Industrial application development exploiting IoT Vision and Model Driven

- programming. In *2015 8th International Conference on Intelligence in Next Generation Networks*, 2015; pp 168-175.
144. Whittle, J.; Hutchinson, J.; Rouncefield, M., The state of practice in model-driven engineering. *Software, IEEE* **2014**, *31* (3), 79-85.
145. Kroiss, C.; Bures, T., Logic-based modeling of information transfer in cyber-physical multi-agent systems. *Future Generation Computer Systems-the International Journal of Escience* **2016**, *56*, 124-139.
146. Xi, Z.; Ieee, Physically Informed Assertions for Cyber Physical Systems Development and Debugging. *2014 Ieee International Conference on Pervasive Computing and Communications Workshops (Percom Workshops)* **2014**, 181-183.
147. Anier, A.; Vain, J.; Tsiopoulos, L., DTRON: a tool for distributed model-based testing of time critical applications. *Proceedings of the Estonian Academy of Sciences* **2017**, *66* (1), 75-88.
148. Hou, Y.; Zhao, Y.; Wagh, A.; Zhang, L.; Qiao, C.; Hulme, K. F.; Wu, C.; Sadek, A. W.; Liu, X., Simulation-Based Testing and Evaluation Tools for Transportation Cyber-Physical Systems. *Ieee Transactions on Vehicular Technology* **2016**, *65* (3), 1098-1108.
149. Slåtten, V.; Herrmann, P.; Kraemer, F. A., Model-Driven Engineering of Reliable Fault-Tolerant Systems-A State-of-the-Art Survey. *Advances in Computers* **2013**, *91*, 119-205.
150. Solé, M.; Muntés-Mulero, V.; Rana, A. I.; Estrada, G., Survey on Models and Techniques for Root-Cause Analysis. *arXiv preprint arXiv:1701.08546* **2017**.
151. Völter, M.; Stahl, T.; Bettin, J.; Haase, A.; Helsen, S., *Model-driven software development: technology, engineering, management*. John Wiley & Sons: 2013.
152. Brambilla, M.; Cabot, J.; Wimmer, M., Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering* **2012**, *1* (1), 1-182.
153. Kordon, F.; Hugues, J.; Canals, A.; Dohet, A., *Embedded Systems: Analysis and Modeling with SysML, UML and AADL*. John Wiley & Sons: 2013.
154. Wehrmeister, M. A.; Pereira, C. E.; Rammig, F. J., Aspect-oriented model-driven engineering for embedded systems applied to automation systems. *IEEE Transactions on Industrial Informatics* **2013**, *9* (4), 2373-2386.
155. Lee, E. A., The Past, Present and Future of Cyber-Physical Systems: A Focus on Models. *Sensors* **2015**, *15* (3), 4837-4869.
156. Weber, P.; Medina-Oliva, G.; Simon, C.; Iung, B., Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence* **2012**, *25* (4), 671-682.
157. Francalanza, E.; Borg, J.; Constantinescu, C., A knowledge-based tool for designing cyber physical production systems. *Computers in Industry* **2017**, *84*, 39-58.
158. Weyns, D.; Ahmad, T. In *Claims and evidence for architecture-based self-adaptation: a systematic literature review*, European Conference on Software Architecture, Springer: 2013; pp 249-265.

159. Kakousis, K.; Paspallis, N.; Papadopoulos, G. A., A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems* **2010**, *4* (4), 355-389.
160. Roy, S.; Biswas, S.; Chaudhuri, S. S., Nature-inspired swarm intelligence and its applications. *International Journal of Modern Education and Computer Science* **2014**, *6* (12), 55.
161. Mavrovouniotis, M.; Li, C.; Yang, S., A survey of swarm intelligence for dynamic optimization: algorithms and applications. *Swarm and Evolutionary Computation* **2017**, *33*, 1-17.
162. Parpinelli, R. S.; Lopes, H. S., New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation* **2011**, *3* (1), 1-16.
163. Muccini, H.; Sharaf, M.; Weyns, D. In *Self-adaptation for cyber-physical systems: a systematic literature review*, Proceedings of the 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM: 2016; pp 75-81.
164. Chen, Y.; Zhou, J.; Guo, M., A context-aware search system for Internet of Things based on hierarchical context model. *Telecommunication Systems* **2016**, *62* (1), 77-91.
165. Li, Y.; Chai, K. K.; Chen, Y.; Loo, J.; Ieee, *Smart Duty Cycle Control with Reinforcement Learning for Machine to Machine Communications*. 2015; p 1458-1463.
166. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S., Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *arXiv preprint arXiv:1610.00633* **2016**.
167. Toser, Z.; Lorincz, A., The Cyber-Physical System Approach Towards Artificial General Intelligence: The Problem of Verification. In *Artificial General Intelligence*, Bieger, J.; Goertzel, B.; Potapov, A., Eds. 2015; Vol. 9205, pp 373-383.
168. Mi, M.; Zolotov, I. In *Comparison between multi-class classifiers and deep learning with focus on industry 4.0*, 2016 Cybernetics & Informatics (K&I), IEEE: 2016; pp 1-5.
169. Wang, H.; Li, Q.; Yi, F.; Li, Z.; Sun, L., Influential spatial facility prediction over large scale cyber-physical vehicles in smart city. *EURASIP Journal on Wireless Communications and Networking* **2016**, *2016* (1), 103.
170. Varshney, K. R.; Alemzadeh, H., On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products. *arXiv preprint arXiv:1610.01256* **2016**.
171. Maier, A. In *Online passive learning of timed automata for cyber-physical production systems*, Industrial Informatics (INDIN), 2014 12th IEEE International Conference on, IEEE: 2014; pp 60-66.
172. Ruchkin, I.; Samuel, S.; Schmerl, B.; Rico, A.; Garlan, D. In *Challenges in physical modeling for adaptation of cyber-physical systems*, Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on, IEEE: 2016; pp 210-215.
173. Macias-Escriba, F. D.; Haber, R.; del Toro, R.; Hernandez, V., Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications* **2013**, *40* (18), 7267-7279.

174. Moreno, G. A.; Cámara, J.; Garlan, D.; Schmerl, B. In *Proactive self-adaptation under uncertainty: A probabilistic model checking approach*, Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ACM: 2015; pp 1-12.
175. Seshia, S. A.; Hu, S.; Li, W.; Zhu, Q., Design Automation of Cyber-Physical Systems: Challenges, Advances, and Opportunities. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2016**.
176. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D., Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* **2016**.
177. Seshia, S. A.; Sadigh, D.; Sastry, S. S., Towards verified artificial intelligence. *arXiv preprint arXiv:1606.08514* **2016**.
178. Khan, F.; Rathnayaka, S.; Ahmed, S., Methods and models in process safety and risk management: Past, present and future. *Process Safety and Environmental Protection* **2015**, 98, 116-147.
179. Vose, D., *Risk analysis: a quantitative guide*. John Wiley & Sons: 2008.
180. Weyns, D.; Iftikhar, M. U.; De La Iglesia, D. G.; Ahmad, T. In *A survey of formal methods in self-adaptive systems*, Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering, ACM: 2012; pp 67-79.
181. Babaei, R.; Babamir, S. M., Runtime verification of service-oriented systems: a well-rounded survey. *International Journal of Web and Grid Services* **2013**, 9 (3), 213-267.
182. Singh, I.; Lee, S.-W. In *Self-adaptive requirements for intelligent transportation system: A case study*, Information and Communication Technology Convergence (ICTC), 2017 International Conference on, IEEE: 2017; pp 520-526.
183. Camilli, M.; Gargantini, A.; Scandurra, P., Zone-based formal specification and timing analysis of real-time self-adaptive systems. *Science of Computer Programming* **2018**, 159, 28-57.
184. Fu, Y.; Shuvo, M. H.; Ieee, An Approach to Analyzing Adaptive Intelligent Vehicle System Using SMT Solver. *2016 International Conference on Control, Decision and Information Technologies (Codit)* **2016**, 313-319.
185. Bersani, M. M.; García-Valls, M. In *The cost of formal verification in adaptive CPS. An example of a virtualized server node*, High Assurance Systems Engineering (HASE), 2016 IEEE 17th International Symposium on, IEEE: 2016; pp 39-46.
186. Klös, V.; Göthel, T.; Glesner, S., Comprehensible and dependable self-learning self-adaptive systems. *Journal of Systems Architecture* **2018**, 85, 28-42.
187. Cámara, J.; Moreno, G. A.; Garlan, D. In *Stochastic game analysis and latency awareness for proactive self-adaptation*, Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, ACM: 2014; pp 155-164.
188. Koh, Y.; Her, H.; Yi, K.; Kim, K., Integrated Speed and Steering Control Driver Model for Vehicle-Driver Closed-Loop Simulation. *IEEE Trans. Vehicular Technology* **2016**, 65 (6), 4401-4411.
189. Li, J.; Zhang, Y.; Chen, Y.; Ieee, *A Self-Adaptive Traffic Light Control System Based on Speed of Vehicles*. 2016; p 382-388.

- 2397 190. Perez-Palacin, D.; Mirandola, R. In *Uncertainties in the modeling of self-adaptive*
2398 *systems: a taxonomy and an example of availability evaluation*, Proceedings of the 5th
2399 ACM/SPEC international conference on Performance engineering, ACM: 2014; pp 3-14.
- 2400 191. Esfahani, N.; Kouroshfar, E.; Malek, S. In *Taming uncertainty in self-adaptive*
2401 *software*, Proceedings of the 19th ACM SIGSOFT symposium and the 13th European
2402 conference on Foundations of software engineering, ACM: 2011; pp 234-244.
- 2403 192. Trapp, M.; Schneider, D., Safety assurance of open adaptive systems—a survey. In
2404 *Models@run. time*, Springer: 2014; pp 279-318.
- 2405 193. McGee, E. T.; McGregor, J. D. In *Using dynamic adaptive systems in*
2406 *safety-critical domains*, Proceedings of the 11th International Symposium on Software
2407 Engineering for Adaptive and Self-Managing Systems, ACM: 2016; pp 115-121.
- 2408 194. Bozhinoski, D. In *Managing safety and adaptability in Mobile Multi-Robot*
2409 *systems*, Proceedings of the 11th International ACM SIGSOFT Conference on Quality of
2410 Software Architectures, ACM: 2015; pp 135-140.
- 2411 195. Klös, V.; Göthel, T.; Glesner, S. In *Formal Models for Analysing Dynamic*
2412 *Adaptation Behaviour in Real-Time Systems*, Foundations and Applications of Self*
2413 Systems, IEEE International Workshops on, IEEE: 2016; pp 106-111.
- 2414 196. Zeller, M.; Prehofer, C. In *Timing constraints for runtime adaptation in real-time,*
2415 *networked embedded systems*, Software Engineering for Adaptive and Self-Managing
2416 Systems (SEAMS), 2012 ICSE Workshop on, IEEE: 2012; pp 73-82.
- 2417 197. Tan, L.; Du, C.; Dong, Y., Control-Performance-Driven Period and Deadline
2418 Selection for Cyber-Physical Systems. *2015 10th Asian Control Conference (Ascc)* **2015**.
- 2419 198. Kim, J.; Kim, H.; Lakshmanan, K.; Rajkumar, R. R. In *Parallel scheduling for*
2420 *cyber-physical systems: Analysis and case study on a self-driving car*, Proceedings of the
2421 ACM/IEEE 4th International Conference on Cyber-Physical Systems, ACM: 2013; pp
2422 31-40.
- 2423 199. Kritikakou, A.; Marty, T.; Roy, M., DYNASCORE: DYNAmic Software
2424 CONTroller to increase REsource utilization in mixed-critical systems. *ACM Transactions*
2425 *on Design Automation of Electronic Systems (TODAES)* **2017**, 23 (2), 13.
- 2426 200. Wu, S.-y.; Zhang, P.; Li, F.; Gu, F.; Pan, Y., A hybrid discrete particle swarm
2427 optimization-genetic algorithm for multi-task scheduling problem in service oriented
2428 manufacturing systems. *Journal of Central South University* **2016**, 23, 421-429.
- 2429 201. Deb, K., Multi-objective optimization using evolutionary algorithms: an
2430 introduction. *Multi-objective evolutionary optimisation for product design and*
2431 *manufacturing* **2011**, 1-24.
- 2432 202. Coello, C. C.; Dhaenens, C.; Jourdan, L., *Advances in multi-objective nature*
2433 *inspired computing*. Springer: 2009; Vol. 272.
- 2434 203. Zhang, K.; Zhang, D.; de La Fortelle, A.; Wu, X.; Grégoire, J., State-driven
2435 priority scheduling mechanisms for driverless vehicles approaching intersections. *IEEE*
2436 *Transactions on Intelligent Transportation Systems* **2015**, 16 (5), 2487-2500.

204. Bambagini, M.; Marinoni, M.; Aydin, H.; Buttazzo, G., Energy-Aware Scheduling for Real-Time Systems: A Survey. *Acm Transactions on Embedded Computing Systems* **2016**, *15* (1).
205. Shepilov, Y.; Pavlova, D.; Kazanskaia, D., Multithreading MAS Platform for Real-Time Scheduling. *International Journal of Software Innovation* **2016**, *4* (1), 48-60.
206. Jiang, J.-M.; Zhu, H.; Li, Q.; Zhao, Y.; Zhao, L.; Zhang, S.; Gong, P.; Hong, Z., Analyzing Event-Based Scheduling in Concurrent Reactive Systems. *ACM Transactions on Embedded Computing Systems (TECS)* **2015**, *14* (4), 86.
207. Ruiz-Arenas, S.; Horváth, I.; Mejía-Gutiérrez, R.; Opiyo, E., Towards the maintenance principles of cyber-physical systems. *Strojniški vestnik-Journal of Mechanical Engineering* **2014**, *60* (12), 815-831.
208. Kans, M.; Galar, D.; Thaduri, A., Maintenance 4.0 in Railway Transportation Industry. In *Proceedings of the 10th World Congress on Engineering Asset Management*, Koskinen, K. T.; Kortelainen, H.; Aaltonen, J.; Uusitalo, T.; Komonen, K.; Mathew, J.; Laitinen, J., Eds. 2016; pp 317-331.
209. Bukowski, L., System of systems dependability—Theoretical models and applications examples. *Reliability Engineering & System Safety* **2016**, *151*, 76-92.
210. Mittal, S.; Vetter, J. S., A survey of techniques for modeling and improving reliability of computing systems. *IEEE Transactions on Parallel and Distributed Systems* **2016**, *27* (4), 1226-1238.
211. Cecati, C., A survey of fault diagnosis and fault-tolerant techniques—part II: fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS* **2015**.
212. Muhammed, T.; Shaikh, R. A., An analysis of fault detection strategies in wireless sensor networks. *Journal of Network and Computer Applications* **2017**, *78*, 267-287.
213. Chouikhi, S.; El Korbi, I.; Ghamri-Doudane, Y.; Saidane, L. A., A survey on fault tolerance in small and large scale wireless sensor networks. *Computer Communications* **2015**, *69*, 22-37.
214. Yang, W.; Liu, Y.; Xu, C.; Cheung, S.-C., A survey on dependability improvement techniques for pervasive computing systems. *Science China Information Sciences* **2015**, *58* (5), 1-14.
215. Li, F.; Shi, P.; Wang, X.; Agarwal, R., Fault detection for networked control systems with quantization and Markovian packet dropouts. *Signal Processing* **2015**, *111*, 106-112.
216. Xia, J.; Jiang, B.; Zhang, K.; Xu, J., Robust Fault Diagnosis Design for Linear Multiagent Systems with Incipient Faults. *Mathematical Problems in Engineering* **2015**.
217. Esfahani, P. M.; Lygeros, J., A tractable fault detection and isolation approach for nonlinear systems with probabilistic performance. *IEEE Transactions on Automatic Control* **2016**, *61* (3), 633-647.
218. Reppa, V.; Polycarpou, M. M.; Panayiotou, C. G., Distributed Sensor Fault Diagnosis for a Network of Interconnected Cyber-Physical Systems. *Ieee Transactions on Control of Network Systems* **2015**, *2* (1), 11-23.

- 2479 219. Dong, H.; Wang, Z.; Ding, S. X.; Gao, H., A Survey on Distributed Filtering and
2480 Fault Detection for Sensor Networks. *Mathematical Problems in Engineering* **2014**.
- 2481 220. Warriach, E. U.; Tei, K. In *Fault detection in wireless sensor networks: A machine*
2482 *learning approach*, Computational Science and Engineering (CSE), 2013 IEEE 16th
2483 International Conference on, IEEE: 2013; pp 758-765.
- 2484 221. Krishnamurthy, S.; Sarkar, S.; Tewari, A. In *Scalable anomaly detection and*
2485 *isolation in cyber-physical systems using bayesian networks*, ASME 2014 Dynamic
2486 Systems and Control Conference, American Society of Mechanical Engineers: 2014; pp
2487 V002T26A006-V002T26A006.
- 2488 222. Sanislav, T.; Merza, K.; Mois, G.; Miclea, L. In *Cyber-physical system*
2489 *dependability enhancement through data mining*, Automation, Quality and Testing,
2490 Robotics (AQTR), 2016 IEEE International Conference on, IEEE: 2016; pp 1-5.
- 2491 223. Khalid, Z.; Fisal, N.; Rozaini, M., A Survey of Middleware for Sensor and
2492 Network Virtualization. *Sensors* **2014**, *14* (12), 24046-24097.
- 2493 224. Zhou, P.; Zuo, D.-C.; Hou, K.-M.; Zhang, Z.; Shi, H.-L., A Light-weight
2494 Multilevel Recoverable Container for Event-driven System: A Self-healing CPS Approach.
2495 **2017**.
- 2496 225. Ahn, S.; Yoo, C.; Lee, S.; Lee, H.; Kim, S. J., Implementing virtual platform for
2497 global-scale cyber physical system networks. *International Journal of Communication*
2498 *Systems* **2015**, *28* (13), 1899-1920.
- 2499 226. Khan, I.; Belqasmi, F.; Glitho, R.; Crespi, N.; Morrow, M.; Polakos, P., Wireless
2500 Sensor Network Virtualization: A Survey. *Ieee Communications Surveys and Tutorials*
2501 **2016**, *18* (1), 553-576.
- 2502 227. Dettoni, F.; Lung, L. C.; Luiz, A. F., Using Virtualization Technology for
2503 Fault-Tolerant Replication in LAN. In *New Results in Dependability and Computer Systems*,
2504 Zamojski, W.; Mazurkiewicz, J.; Sugier, J.; Walkowiak, T.; Kacprzyk, J., Eds. 2013; Vol.
2505 224, pp 131-140.
- 2506 228. Quattrocio, W.; Caldarelli, G.; Scala, A., Self-Healing Networks: Redundancy
2507 and Structure. *Plos One* **2014**, *9* (2).
- 2508 229. Hazra, A.; Dasgupta, P.; Chakrabarti, P. P., Formal assessment of reliability
2509 specifications in embedded cyber-physical systems. *Journal of Applied Logic* **2016**, *18*,
2510 71-104.
- 2511 230. Cui, Y.; Lane, J.; Voyles, R.; Krishnamoorthy, A. In *A new fault tolerance method*
2512 *for field robotics through a self-adaptation architecture*, Safety, Security, and Rescue
2513 Robotics (SSRR), 2014 IEEE International Symposium on, IEEE: 2014; pp 1-6.
- 2514 231. Warriach, E. U.; Ozcelebi, T.; Lukkien, J. J. In *Fault-prevention in smart*
2515 *environments for dependable applications*, Self-Adaptive and Self-Organizing Systems
2516 (SASO), 2014 IEEE Eighth International Conference on, IEEE: 2014; pp 183-184.
- 2517 232. Zheng, J.; Wang, P.; Li, C.; Mouftah, H. T. In *An Efficient Fault-Prevention*
2518 *Clustering Protocol for Robust Underwater Sensor Networks*, Communications, 2008.
2519 ICC'08. IEEE International Conference on, IEEE: 2008; pp 2802-2807.

233. Cooray, D.; Kouroshfar, E.; Malek, S.; Roshandel, R., Proactive self-adaptation for improving the reliability of mission-critical, embedded, and mobile software. *IEEE Transactions on Software Engineering* **2013**, *39* (12), 1714-1735.
234. Yano, I. H.; Oliveira, V. C.; de Mello Fagotto, E. A.; Mota, A. D. A.; Mota, L. T. M., Predicting battery charge depletion in wireless sensor networks using received signal strength indicator. *Journal of Computer Science* **2013**, *9* (7), 821-826.
235. Warriach, E. U.; Ozcelebi, T.; Lukkien, J. J. In *A Comparison of Predictive Algorithms for Failure Prevention in Smart Environment Applications*, Intelligent Environments (IE), 2015 International Conference on, IEEE: 2015; pp 33-40.
236. Németh, E.; Lakner, R.; Hangos, K. M.; Cameron, I. T., Prediction-based diagnosis and loss prevention using qualitative multi-scale models. *Information Sciences* **2007**, *177* (8), 1916-1930.
237. Lakner, R.; Németh, E.; Hangos, K. M.; Cameron, I. T. In *Multiagent realization of prediction-based diagnosis and loss prevention*, International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer: 2006; pp 70-80.
238. Priesterjahn, C.; Steenken, D.; Tichy, M. In *Component-based timed hazard analysis of self-healing systems*, Proceedings of the 8th workshop on Assurances for self-adaptive systems, ACM: 2011; pp 34-43.
239. Salfner, F.; Lenk, M.; Malek, M., A survey of online failure prediction methods. *ACM Computing Surveys (CSUR)* **2010**, *42* (3), 10.
240. Verma, A. K.; Srividya, A.; Ramesh, P., A systemic approach to integrated E-maintenance of large engineering plants. *International Journal of Automation and Computing* **2010**, *7* (2), 173-179.
241. Sleptchenko, A.; Johnson, M. E., Maintaining Secure and Reliable Distributed Control Systems. *INFORMS Journal on Computing* **2014**, *27* (1), 103-117.
242. Wan, K.; Alagar, V. In *Achieving dependability of cyber physical systems with autonomic covering*, Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on, IEEE: 2014; pp 139-145.
243. Asplund, F., The future of software tool chain safety qualification. *Safety Science* **2015**, *74*, 37-43.