2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)

Modelling and Analysing Resilient Cyber-Physical Systems

Amel Bennaceur, Carlo Ghezzi, Kenji Tei, Timo Kehrer, Danny Weyns, Radu Calinescu, Schahram Dustdar, Zhenjiang Hu, Shinichi Honiden, Fuyuki Ishikawa, Zhi Jin, Jeffrey Kramer, Marin Litoiu, Michele Loreti, Gabriel A. Moreno, Hausi A. Müller, Laura Nenzi, Bashar Nuseibeh, Liliana Pasquale, Wolfgang Reisig, Heinz Schmidt, Christos Tsigkanos, Haiyan Zhao
Shonan Seminar 118 Participants
Email: shonan_meeting_118@nii.ac.jp

***Abstract*—From smart buildings to medical devices to smart nations, software systems increasingly integrate computation, networking, and interaction with the physical environment. These systems are known as Cyber-Physical Systems (CPS). While these systems open new opportunities to deliver improved quality of life for people and reinvigorate computing, their engineering is a difficult problem given the level of heterogeneity and dynamism they exhibit. While progress has been made, we argue that complexity is now at a level such that existing approaches need a major re-think to define principles and associated techniques for CPS. In this paper, we identify research challenges when modelling, analysing and engineering CPS. We focus on three key topics: theoretical foundations of CPS, self-adaptation methods for CPS, and exemplars of CPS serving as a research vehicle shared by a larger community. For each topic, we present an overview and suggest future research directions, thereby focusing on selected challenges. This paper is one of the results of the Shonan Seminar 118 on *Modelling and Analysing Resilient Cyber-Physical Systems*, which took place in December 2018.**

I. Introduction

The ultimate goal of any software system is to support individuals and groups in their social and professional endeavours. This is ever more important today that software permeates every aspect of our lives. From smart buildings to medical devices to smart nations, software systems increasingly integrate computation, networking, and interaction with the physical environment. These systems are known as Cyber-Physical Systems (CPS). The National Institute of Standards and Technology (NIST) define them as follows: "*Cyber physical systems are hybrid networked cyber and engineered physical elements co-designed to create adaptive and predictive systems for enhanced performance. Performance metrics include safety and security, reliability, agility and stability, efficiency and sustainability, privacy.*" [1].

The need for self-management and self-adaptation is inherent in CPS: they are long-lived, continuously running systems that interact with the environment and humans in ways that can hardly be fully anticipated at design time and continuously evolve at runtime. In other words, CPS must be resilient, that is able to self-adapt to deal with change. Yet, existing software engineering methods often focus on sanitised environments, abstracting away many details including those related to the physical properties of the environment. Theory, methodology, and tools for the systematic design and engineering of CPS are yet to be defined.

First, theories are crucial to understand the interplay between the physical and the digital worlds. Typically, the changing topology of space in which computations are embedded needs to be understood during design and managed properly during operation. Furthermore, as many of those systems are safety critical, rigorous modelling and analysis are necessary to provide guarantees about the overall behaviour of the CPS. This rigorous design is often challenged by the differences in nature of the components of these systems, including discrete-time computation components as well as continuous-time physical components. Hence integration is made more challenging and so is planning and controlling of the emergent behaviour of multiple such hybrid systems. Furthermore, many transversal issues such as security and adaptation are made more difficult due to the inherent uncertainty of the physical environment, and the incompleteness of any model thereof.

The timeliness of this topic is illustrated through (a) an increasing number of papers and surveys in the domain such as the systematic survey by Muccini *et al.* [2] which focuses specifically on architectural adaptation for CPS, (b) some dedicated workshops such as SEsCPS [3] which focuses on smart CPS, (c) standardisation effort such as the NIST Cyber-Physical Systems Program [1], (d) multiple organised working seminars such as SENCPS [4] which explores synergies between software engineering and CPS, and (e) multiple government initiatives such as NSF Cyber-Physical Systems Virtual Organization [5] and EU CyPhERS (Cyber-Physical European Roadmap & Strategy) [6].

This paper reports on the outcome of a Shonan seminar that aimed to reflect on both the theoretical and practical underpinning of resilient CPS. The key topics investigated were: (i) Theory of Resilient CPS (ii) Design and Engineering of Resilient CPS, and (iii) Applications and Exemplars for CPS. These topics require the interaction of many areas of expertise in embedded systems, verification and simulation, and spatial reasoning besides those in software engineering and adaptive software systems. In the following, we summarise the main points discussed during the seminar.

Section II focuses on theory and highlights the need to rethink the modelling of software to account for ecosystems of

978-1-7281-3368-3/19/$31.00 ©2019 IEEE
DOI 10.1109/SEAMS.2019.00018

CPS, to redefine assurances in terms of equilibrium, and how these theoretical concepts can be taught. Section III moves to the engineering space and focuses on self adaptation. It highlights the need for decentralised control of ecosystems of CPS and highlights human behaviour aspects. Section IV motivates and presents a feature-based classification scheme that serves as an instrument to organise and structure CPS exemplars. Finally, Section V concludes the paper.

II. Foundation

A. Motivation

Resilient CPS involve rethinking design and engineering with a major focus on composition and dynamic environments. One possible way to capture those aspects is considering *ecosystems* that compose software platforms as well as communities of users [7].

The rigorous analysis of CPS requires models that represent heterogeneous aspects of CPS across different layers of the technology stack—from the physical, sensor and actuator layer, to communication and middleware, up to application layer. Models may be required across tiers of the CPS, to represent heterogeneous types of software, from user applications to supporting services and back-end storage. These are inherently multi-faceted and typically belong to different disciplines (e.g., physical, communication, software, social). An important challenge is then how to align the abstractions of these heterogeneous models into a unified representation that allows for reasoning and supporting adaptation decisions.

While rigorously representing CPS is difficult, their composition, analysis, and adaptive control are even more challenging [8]. In particular adaptive control of CPS is challenging due to their inherent hybrid nature. On the one hand, discrete-time control focuses on functional requirements, deals with composition but require complete knowledge of the environment. On the other hand, continuous-time control focuses on quantitative requirements, adapt to perturbations in the environment but does not support composition and concurrency. Defining appropriate *assurance* properties and methods for CPS is essential.

Given the diversity of techniques and methods that are foundational for modelling and analysing CPS, curricula that prepare and train a skilled workforce should reflect this diversity and multidisciplinarity.

B. Ecosystems

The choice of the environment depends on the scale of the system at hand: how a CPS is defined depends on the scope and the context. Figure 1 gives an example on how we can model an automotive system at four different levels of granularity. For each level of granularity, the notion of environment is defined with respect to the chosen system. For example, while modelling the engine, the environment is made up of the other components of the car. When modelling a car, then other cars compose the environment. When designing a platoon, then the transport infrastructure can represent the environment. Finally, when considering a smart city as a CPS, then the environment may include other cities.

The scope and goals of those ecosystems need to be well understood in order for the impact of collaboration and interconnection to be specified rather than just incurred. Understanding, yet alone controlling, emergent collaborations between communities of users and CPS, and the theory and processes for understanding them are still to be defined.

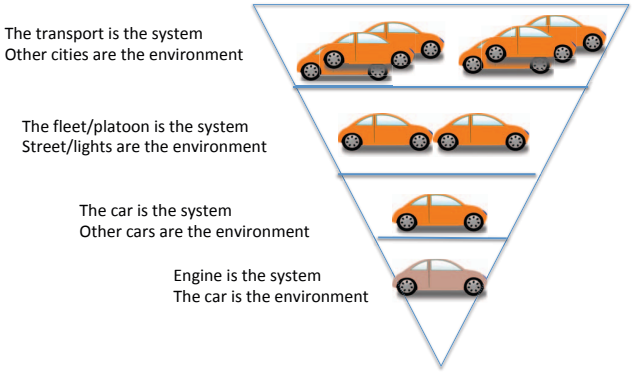

Fig. 1. Illustrating CPS ecosystems on transport

C. Assurance

Resilience has been defined as the persistence of dependability while facing change [9], and often understood as the ability of the system to return to a viable zone/stability [10] while avoiding Zeno behaviour, i.e. the system undergoing an unbounded number of discrete transitions in a finite and bounded length of time. The classic notion of satisfaction is insufficient to describe properties of such behaviour. Therefore, we consider the notion of *equilibrium* as a new form of satisfaction. The idea is that the system maintains a behaviour within its multidimensional viability zone rather than satisfy a property in the case of perturbations. Moreover, the system actively monitors whether it is in its normal viability zone and is able to bring it back within if it ventures outside. After returning or healing, the system can potentially be stronger and so even the bounds can change leading to contextual viability zones [11]. Different definitions of this notion can lead to different interpretations and requirements.

For self-adaptive CPS, assurances must also consist of *comprehensive evidence* (obtained through modelling and simulation, testing, formal verification, compliance with established practices, etc.) that the CPS can safely achieve the goals of their intended application in the physical environment in which they operate. Given the heterogeneity and distributed nature of many CPS and the complexity of their goals, devising this comprehensive body of evidence represents a major challenge that is not fully addressed by existing approaches.

A further challenge in the provision of assurances for CPS self-adaptation is the need to *integrate assurance evidence* from all stages of the CPS lifecycle. Assurance cases for CPS must combine development-time evidence from the CPS design, implementation and verification with runtime evidence that they continue to safely achieve their goals during self-adaptation. Dynamic safety cases have been used to tackle this

challenge for self-adaptive software [12], but extending their applicability to CPS requires significant additional research due to the physical aspects of these systems and of their goals.

Modelling and reasoning about spatio-temporal properties is also important. *Cyber-physical spaces* [13] are composite models integrating human agents, computational and physical aspects of systems. Formal languages such as spatio-temporal logics [14], [15] can be used to describe, verify, and test complex properties where the spatial and temporal part are intrinsically connected and influence each other. Furthermore, they provide efficient monitoring procedures to verify the property and they deal with changes in spatial configuration.

D. Education

The multifaceted nature of designing and engineering CPS raises multiple questions on how to educate students with those foundational concepts in CPS in order to create and maintain a skilled workforce to support the design, engineering, deployment, and operation of future CPS. CPS engineers, scientists and developers not only need strong backgrounds in CPS foundations, but also significant knowledge in relevant application domains. The cross-cutting and rapidly evolving application of sensing, actuation, control, communication and computing presents significant challenges for industry, academia and governments. Existing engineering and computer science programs are challenged in teaching the comprehensive skill set required for a successful career in the CPS realm [16]. The software engineering community has made tremendous strides in designing and operating highly dynamical software systems by developing methods and techniques to deal with CPS uncertainty and resilience at runtime as well as standardise and distribute CPS components and services effectively. It is high time to inject these innovations into computing and software curricula which still largely concentrate on design-time aspects of, for example, requirements, models and V&V (Verification and Validation). Digital control, which integrates discrete and continuous mathematics, is central to CPS. On the one hand, computer science and software engineering programs need digital control courses; on the other hand, traditional engineering programs need to include software engineering courses. Designing CPS contents involves a careful balancing of physical and cyber aspects as well as CPS application knowledge [17]. While adding CPS courses, options or degree programs is extremely challenging due to the many competing forces, trained CPS students are needed in industry to harvest CPS rich economic opportunities.

III. Engineering Self-Adaptation for Resilient CPS

A. Motivation

CPS must handle high levels of dynamicity and uncertainty. This is due to factors that include workload variation, interactions with human users and operators, regular goal changes, and components joining and leaving the CPS. As such, the software controlling the CPS operation must manage its dynamicity and uncertainty, using self-adaptation to ensure that the system behaviour stays within the bounds defined by its goals. For CPS used in safety-critical applications, these goals often specify strict safety, dependability and performance requirements. Accordingly, the CPS control software must also provide assurances guaranteeing the system compliance with these requirements. While the features we mentioned so far are common to most types of self-adaptive systems, several distinguishing characteristics of CPS further increase the challenges associated with the engineering of their control software. First, the heterogeneity of the CPS components and of their sensors and actuators (vertically across the technology stack, and horizontally across different components and subsystems) greatly increases the complexity of the control software. Second, the distributed deployment of most CPS, often with only unreliable, high-latency or low-bandwidth communication affordable between components, precludes the maintenance of up-to-date global system models. Third, even when such global models can be assembled and kept up to date, they are typically too large to be analysed efficiently and to support timely reasoning about the CPS. Fourth, many CPS are assembled through the integration of components owned by different organisations. Last but not least, the constraints and optimisation criteria specified by CPS goals refer not only to computational aspects such as throughput and task ordering, but also to physical aspects of the system components.

This unique combination of characteristics is responsible for multiple open challenges in developing self-adaptation methods and software for resilient CPS. In the remainder of this section, we summarise four of these open challenges that we expect to drive future research in this area.

B. Control software decentralisation

For the numerous CPS for which system-level modelling and analysis are unfeasible, or the system components are owned by multiple organisations, the control software needs to be decentralised. Examples of such CPS include many Internet of Things (IoT) systems, unmanned-vehicle CPS, and smart e-health CPS. For instance, to support multiple tenants and increase the scale of the IoT system presented in [18], the control software necessarily needs to be decentralised to enable local decision-making while keeping the energy consumption of battery-powered modes within bounds. As another example, consider the CPS of unmanned underwater vehicles from [19], the driving factors for decentralising the control software are the efficiency of modelling and analysis, and ensuring the mission goals regardless of the inherent restrictions of communication under water. Finally, in a smart e-health CPS as the one presented in [20], different parts of the systems have different owners that may be unable to share all information (e.g., for security or privacy reasons); hence, autonomy of subsystems and decentralising the control software is imperative.

In summary, decentralising self-adaptation enables dealing with multiple owners and autonomy of CPS components, and inherent distribution and restrictions of resources. However, successfully decentralising the CPS control software is neither a panacea nor without its costs. We highlight four implications or

potential drawbacks, together with their associated challenges and starting points for addressing them.

As CPS are often long-living systems that organically grow, decentralisation of control software can serve as an enabler to support robust and scalable system evolution. However, this raises the challenge of suitable coordination capabilities for entities to join and leave the CPS ecosystem. Agent coordination and protocols [21] could be a starting point for tackling this challenge.

Decentralisation of the CPS control software requires adaptation decisions to be made based on locally available information that are not necessarily altruistic. Consequently, the decisions may be sub-optimal compared to global decision-making. The challenges are then how to measure and quantify the cost of decentralising the control software in terms of loss of decision-making optimality. This cost may then be traded against the degree of decentralisation, e.g., by structuring decision-making for adaptation hierarchically. One source of inspiration to study these challenges is "Price of anarchy" [22], which is a concept from economics and game theory that allows measuring how a system's efficiency degrades as a result of distributed competitive decision making.

Decentralisation of control software may raise trust issues as well. In a decentralised setting, the subsystems of a CPS may be unwilling or unable to share all the information needed for local decisions, e.g., on how to perform re-configurations. A challenge is then how to ensure sufficient trust in the system and how to ensure that no undesired effects emerge from local decisions? Interesting approaches to start tackling this challenge are computational mechanism design and game theory [23].

An important aspect of CPS is incident handling, e.g., due to security or privacy events. An important challenge is then to understand the impact of decentralisation of control software on incident handling. This impact can be considered from two perspectives: on the one hand, detecting incidents may be more difficult due to locality of activities; on the other hand, the effects may be localised, reducing the harm caused by incidents.

C. Adaptive Security for CPS

As CPS span cyber and physical spaces, they are more vulnerable than conventional software systems to attacks [24]. Malicious actors can exploit cyber accessibility to a digital network to gain access to the physical devices connected to the network (e.g., German Still Mill Attack [25]). Malicious actors can also exploit vulnerabilities of physical devices to control them remotely and orchestrate attacks against third party systems and services (e.g., Mirai Attack [26]).

So far security risks arising from the cyber and physical spaces have been assessed separately [27], leading to gaps and vulnerabilities for parts of the system. Thus, traditional risk assessment methods (e.g., CORAS [28]) need to be revised and should consider the extended attack surface brought by the interplay between cyber and physical components in CPS.

Unpredictability, heterogeneity, and scale make it difficult to anticipate how security threats can materialise and what security countermeasures to apply to prevent them. To protect todays CPS, designing static and rigid security solutions is no longer sufficient. CPS should be designed with the capability to self-protect [29], [30], especially when security threats may arise from different spaces.

Existing approaches proposed to develop self-protecting software systems (e.g., [31]) usually can only react to a set of changes (in the system or its operating environment) that are known at design time by enacting a set of pre-defined countermeasures. This would still leave the sub-system to be protected exposed, for example, to attacks targeting new assets or exploiting vulnerabilities brought by changes in the topology (structure and connectivity) of cyber and physical components. Thus it is necessary to develop novel threat analysis and planning techniques to reason about changing security threats and selecting a set of countermeasures that could guarantee assets protection. These techniques should scale by adaptively focusing on the aspects of the CPS that require protection.

D. Models at runtime

The self-adaptation methods used by CPS must efficiently and coherently leverage multiple types of models at runtime. Models used for self-adaptation often capture uncertainties (e.g., in terms of probabilities of properties in the environment), or the models themselves may have uncertainty (e.g., due to sensor noise). Given the heterogeneity of CPS, a challenge is then how to ensure that the runtime models are sufficiently accurate to make timely adaptation decisions. Rephrased from a models@runtime perspective, the question raised by this challenge is: what does causal connection[1] mean for runtime models of CPS, and how can this causality be realised?

As CPS are often large-scale systems and the control software for self-adaptation is decentralised, an important challenge is to decide what information is collected where, what and how is this information shared, and how to ensure that the distributed models used to support decision making for self-adaptation are consistent across the CPS components.

E. Human stakeholders

The self-adaptation methods employed by CPS must provide *relevant and comprehensible information to stakeholders* ranging from users and operators to regulators and the general public [32]. This includes information about the rationale underpinning self-adaptation decisions (e.g., to gain the trust of users, and to enable CPS certification by regulators), and information supporting users and operators in their regular interactions with the system.The adoption and success of many envisaged CPS depend on this challenge being addressed by the research community.

Numerous CPS used in smart cities, e-health, smart transportation and similar applications are complex socio-technical systems. Humans who interact with these CPS are not merely providers of system input and consumers of artefacts produced

[1]Recall that a causal connection refers to the link between the managed system and the model representing it such that whenever a change is made to the model, this change is reified in the system and whenever the system changes, this change is reflected in the corresponding model

by the system. They are first-class *participants in the CPS*, whom the system relies upon for contributions to decision making, to the execution of these decisions, etc. This means that the self-adaptation methods employed by these CPS must consider human participants in all their steps—from the monitoring and analysis of the system and its environment, to the synthesis of adaptation plans and the execution of these plans. While preliminary work and thoughts on self-adaptive systems with "humans in the loop" (e.g., [33], [34]) provide a starting point for tackling this challenge, further research is needed to apply these concepts to CPS with human participants.

Research has emphasised the need for social adaptation, where the software system analyses users' feedback and updates its behaviour to best satisfy the requirements in the given context [35]. In fact with the prevalence of mobile and ubiquitous technology, it is becoming easier to have a better understanding of user preferences, and one can aim to compose both digital and social services [36].

IV. Exemplars

A. Motivation

Good (software) engineering research not only requires methodological, technical and theoretical results, but also convincing evidence that these results are sound [37]. Exemplars are well-suited for validation, studying relevant problems, and as a medium for education. Exemplars have been collected and established in various areas of engineering software-intensive systems, e.g., in requirements engineering [38], software and system evolution [39], software product-line engineering [40], and self-adaptive and self-managing systems [41]. However, to the best of our knowledge there is no structured catalogue or repository of exemplars specifically addressing CPS.

Therefore, our goal is to provide comprehensive information about CPS exemplars that would be otherwise scattered in the literature or restricted to local usage in dedicated laboratories, such as the *Cyber-Physical Systems Laboratory* at the HPI [42] or the *Virtual Experiences Laboratory (VXLab)* at the Royal Melbourne Institute of Technology [43], [44]. The primary target group comprises researchers and educators who can use the collection as a source of information to find the exemplars which fit to their individual needs. We focus on a common classification scheme for characterising the exemplars and a technical infrastructure for collecting these exemplars.

B. Classification Scheme

As mentioned above, collections of exemplars have been established by several research communities. The SEAMS community maintains a catalogue of exemplars for self-adaptive systems, ranging from generic artefacts to specific model problems [41]. Some of these exemplars are specifically addressing CPS and represent a good starting point for our classification scheme. Yet, our goal is to address CPS from a broader perspective, including further qualities besides self-adaptation and -management.

Moreover, exemplars in the SEAMS catalogue are mainly described in an unstructured way using natural language. While this has the advantage that providing new exemplars is easy, searching for an exemplar offering specific characteristics can be difficult and tedious. Therefore, we propose a more detailed classification scheme that enables structured descriptions of CPS exemplars amenable to (semi-)automated search. This scheme should allow one to a) characterise the general kind of CPS represented by an exemplar as well as b) characterise a specific exemplar itself.

a) Characterising the kind of CPS represented by an exemplar: To characterise the general kind of CPS represented by an exemplar, we rely on techniques that are primarily known from the field of software product-line engineering, particularly the use of feature models [45]. These have proven well-suited for structuring a domain of interest. The idea is that the features including their inter-relations formally capture the variation points of the set of conceivable exemplars, while the kind of system represented by a specific exemplar is precisely characterised by a valid configuration of the feature model. Besides formally documenting the main variation points of a CPS, such a feature model also provides a common yet high-level terminology for CPS, which is of increasing importance given its interdisciplinary nature. Our aim is not to come up with an exhaustive taxonomy or ontology, but with a feature model which is generic enough to classify any kind of CPS of interest and specifically tailored for our purpose of describing exemplars. An excerpt of an early version of our feature model is shown in Figure 2.

A first variation point to do a high-level characterisation is the *Domain* in which a CPS is intended to operate. Some typical domains are *Healthcare*, *Transportation* or *Food Security*. Another high-level yet distinguishing feature is whether a CPS emphasises the role of the *Human* interacting with the system or not.

In addition, there are a number of cross-cutting features which, regardless of the particular domain and regardless of whether the CPS emphasises human interaction, are interesting for validating a broad range of generic methods as:

Qualities. Since we are specifically addressing the analysis of CPS, one important variation point pertains the *Qualities* which we expect to be exposed by a particular kind of CPS. Qualities of interest include *Dependability* properties such as *Safety*, *Security* and *Privacy*. *Self-Adaptivity* leads to improvements in dependability. Specifically, considering our example dependability properties, *Self-Healing* and *Self-Protection* refer to the automatic detection of failures and attacks as well as their subsequent correction and suppression, respectively.

Distribution. CPS are highly distributed systems by definition. However, we may distinguish whether *Distribution* is only *Virtual* or also *Spatial*. The former reflects the classical notion of a distributed system where computational entities are distributed and connected over some network structure, while the latter applies to CPS which are designated to be operated in a larger spatial environment such as smart buildings or cities.

Evolution. Another aspect which is of particular interest for

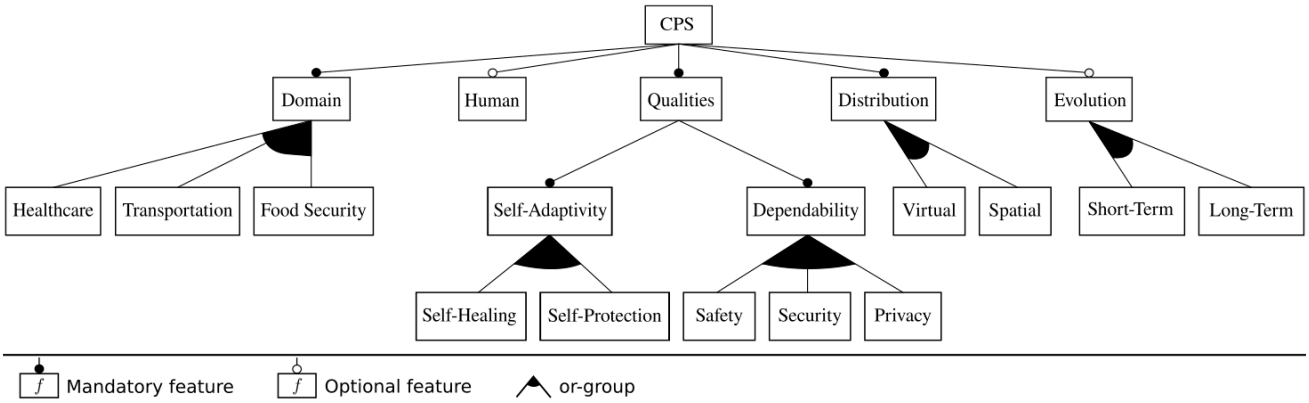

Fig. 2. Excerpt of an early version of our feature-based classification scheme for characterising the general kind of CPS represented by an exemplar.

various analysis methods is *Evolution*, where we distinguish among *Short-Term* evolution and *Long-Term evolution*. Short-term evolution means that the system operates in a highly dynamic environment undergoing continuous changes, while long-term evolution stresses the fact that a system is intended to be operated for a long period of time.

For example, let us consider two concrete CPS exemplars from the SEAMS catalogue: The Automated Traffic Routing Problem (ATRP) [46] and an IoT-based ecosystem to support nutrition called "Feed me, Feed me" (FmFm) [47]. According to our feature-based classification, both systems have a set of common and individual features. While stemming from different domains, namely *Transportation* and *Food Security*, both systems share a highly dynamic nature (*Short-Term*) and must deal with frequent changes and uncertainty (*Adaptivity*). Concerning further qualities, FmFm produces vast quantities of personal data which demands for robust protection mechanisms (*Security* and *Privacy*), while *Safety* is one of the primary concerns for ATRP. Moreover, ATRP clearly operates in a *Spatial* environment, while this dimension of distribution is of minor importance for FmFm. However, in contrast to ATRP, FmFm puts forward the shared control and partial automation between the software system and its users in the social dimension (*Human*).

b) Characterisation of a specific exemplar: In addition to the characterisation of the kinds of systems represented by an exemplar, exemplars shall be further characterised by collecting meta-data that are specific to an exemplar instance.

Generic Meta-data include but are not limited to, e.g., literature references where the exemplar has been used, which kinds of artefacts are available for the exemplar, and, if available, a literature reference to where the exemplar has been originally published as well as further pointers where to find more detailed information about the exemplar.

In order to evaluate the scalability of a method, researchers might also be interested in the *Size* of an exemplar. For our classification scheme, we propose to use a purely qualitative classification into *Small-*, *Medium-* and *Large*-scaled exemplars.

Optionally, an exemplar may also be intended for serving a particular *Purpose*. Typical purposes are to drive and communicate individual research advances, to compare and contrast alternative approaches, to establish research agendas, and, ultimately, to lead to advances in practices of developing and operating certain kinds of CPSs. This characterisation can be useful since, as argued in [38], there are interferences between these different purposes of exemplars, and an exemplar suited to serve one purpose is not necessarily suited to serve another.

V. Conclusion

As CPS increasingly permeate every aspect of modern systems, it is important to define the theoretical and practical foundation for modelling, analysing, and adapting them. This paper summarises some directions towards this goal.

The theoretical foundation involves modelling ecosystems at different levels of granularity and focusing on their composition. It also involves a rich notion of assurance, that not only focuses on the satisfaction of well-specified requirements but also on defining equilibrium and contextual satisfaction.

From an engineering perspective, more decentralisation is necessary to account for the inherently distributed nature of these systems. Models at runtime allows for accommodating the continuous change in the environments of CPS. Accounting for humans as an integral parts of those systems also raises multiple challenges for adaptation. In addition, one cannot forget that with openness comes threats and therefore CPS also need adaptive methods to protect themselves.

Finally, establishing a technical infrastructure for collecting and describing CPS exemplars, and for querying and browsing these exemplars is essential, not only for evaluating research but also for education. Given the multifaceted nature of designing and engineering CPS, creating CPS curricula and courses involves a careful balancing of theoretical and practical knowledge of physical and cyber aspects as well as knowledge of CPS applications.

We believe that CPS hold great promise for research in self-adaptation. They also raise many challenges. This paper gave a summary of these challenges and we invite other researchers, educators and practitioners to collaborate with us in addressing them.

ACKNOWLEDGMENT

The authors would like to thank the staff of Shonan Village for their valuable support. We acknowledge SFI grant 13/RC/2094 and EPSRC support.

REFERENCES

[1] "NIST cyber-physical systems program," http://www.nist.gov/el/cyber-physical-systems, 2019.

[2] H. Muccini, M. Sharaf, and D. Weyns, "Self-adaptation for cyber-physical systems: a systematic literature review," in *Proc. of SEAMS*, 2016.

[3] T. Bures, D. Weyns, B. R. Schmerl, E. Tovar, E. Boden, T. Gabor, I. Gerostathopoulos, P. Gupta, E. Kang, A. Knauss, P. Patel, A. Rashid, I. Ruchkin, R. Sukkerd, and C. Tsigkanos, "Software engineering for smart cyber-physical systems: Challenges and promising solutions," *ACM SIGSOFT Software Engineering Notes*, vol. 42, no. 2, pp. 19–24, 2017.

[4] T. Tamai, H. A. Muller, and B. Nuseibeh, "Software engineering and networked control for smart cyber physical systems," Shonan, Tech. Rep., 2017, https://bit.ly/2GiO7n7.

[5] "Cyber-physical systems virtual organization," https://cps-vo.org, 2019.

[6] "Cyphers - cyber-physical european roadmap and strategy," http://www.cyphers.eu, 2019.

[7] J. Bosch, "Speed, data, and ecosystems: The future of software engineering," *IEEE Software*, vol. 33, no. 1, pp. 82–88, 2016.

[8] J. Sifakis, "System design in the era of iot - meeting the autonomy challenge," in *Proc. of MeTRiD@ETAPS 2018*, 2018, pp. 1–22.

[9] J.-C. Laprie, "From dependability to resilience," in *Proc. of the 38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks*, 2008.

[10] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre, *Viability Theory: New Directions*. Springer, 2011.

[11] R. e. a. de Lemos, "Software engineering for self-adaptive systems: Research challenges in the provision of assurances," in *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 2017, pp. 3–30.

[12] R. Calinescu, D. Weyns, S. Gerasimou, M. U. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Trans. on Soft. Eng.*, vol. 44, no. 11, 2018.

[13] C. Tsigkanos, T. Kehrer, and C. Ghezzi, "Modeling and verification of evolving cyber-physical spaces," in *Proc. of ESEC/FSE*, 2017, pp. 38–48.

[14] E. Bartocci, L. Bortolussi, M. Loreti, and L. Nenzi, "Monitoring mobile and spatially distributed cyber-physical systems," in *Proc. of MEMOCODE*, 2017, pp. 146–155.

[15] P. Herrmann, J. O. Blech, F. Han, and H. W. Schmidt, "A model-based toolchain to verify spatial behavior of cyber-physical systems," *Int. J. Web Service Res.*, vol. 13, no. 1, pp. 40–52, 2016.

[16] H. A. Müller, "The rise of intelligent cyber-physical systems," *IEEE Computer*, vol. 50, no. 12, pp. 7–9, 2017.

[17] J. A. Stankovic, J. W. Sturges, and J. Eisenberg, "A 21st century cyber-physical systems education," *IEEE Computer*, vol. 50, no. 12, 2017.

[18] D. Weyns, M. U. Iftikhar, D. Hughes, and N. Matthys, "Applying architecture-based adaptation to automate the management of internet-of-things," in *Proc. of the 12th European Conference on Software Architecture, ECSA*, 2018, pp. 49–67.

[19] R. Calinescu, S. Gerasimou, and A. Banks, "Self-adaptive software with decentralised control loops," in *Fundamental Approaches to Software Engineering*, A. Egyed and I. Schaefer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 235–251.

[20] N. Li, C. Tsigkanos, Z. Jin, S. Dustdar, Z. Hu, and C. Ghezzi, "Poet: Privacy on the edge with bidirectional data transformations," in *International Conference on Pervasive Computing and Communications*. IEEE Press, 2019.

[21] M. N. Huhns and L. M. Stephens, in *Multiagent Systems*, G. Weiss, Ed. Cambridge, MA, USA: MIT Press, 1999, ch. Multiagent Systems and Societies of Agents, pp. 79–120.

[22] T. Roughgarden, "Selfish routing and the price of anarchy," *MIT Press*, 2005.

[23] R. K. Dash, N. R. Jennings, and D. C. Parkes, "Computational-mechanism design: a call to arms," *IEEE Intelligent Systems*, vol. 18, no. 6, pp. 40–47, Nov 2003.

[24] L. Pasquale, C. Ghezzi, C. Menghi, C. Tsigkanos, and B. Nuseibeh, "Topology aware adaptive security," in *Proc. of SEAMS*, 2014, pp. 43–48.

[25] R. M. Lee, M. J. Assante, and T. Conway, "German Steel Mill Cyber Attack," *Industrial Control Systems*, vol. 30, p. 62, 2014.

[26] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, no. 2, pp. 76–79, 2017.

[27] A. van Cleeff, W. Pieters, R. Wieringa, and F. van Tiel, "Integrated Assessment and Mitigation of Physical and Digital Security Threats: Case Studies on Virtualization," *Information security technical report*, vol. 16, no. 3-4, pp. 142–149, 2011.

[28] F. Den Braber, I. Hogganvik, M. S. Lund, K. Stølen, and F. Vraalsen, "Model-Based Security Analysis in Seven StepsA Guided Tour to the CORAS Method," *BT Technology Journal*, vol. 25, no. 1, pp. 101–117, 2007.

[29] E. Yuan, N. Esfahani, and S. Malek, "A Systematic Survey of Self-Protecting Software Systems," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 4, p. 17, 2014.

[30] A. Bennaceur, T. T. Tun, A. K. Bandara, Y. Yu, and B. Nuseibeh, "Feature-driven mediator synthesis: Supporting collaborative security in the internet of things," *TCPS*, vol. 2, no. 3, pp. 21:1–21:25, 2018.

[31] C. Tsigkanos, L. Pasquale, C. Ghezzi, and B. Nuseibeh, "On the Interplay Between Cyber and Physical Spaces for Adaptive Security," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 3, pp. 466–480, 2018.

[32] S. Dustdar, "Towards building cyber-physical ecosystems of people, processes, and things," in *Proc. of the 1st International Conference on Complex Information Systems, COMPLEXIS*, 2016, p. 11.

[33] J. Cámara, G. A. Moreno, and D. Garlan, "Reasoning about human participation in self-adaptive systems," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE Press, 2015, pp. 146–156.

[34] D. Weyns, N. Bencomo, R. Calinescu, J. Camara, C. Ghezzi, V. Grassi, L. Grunske, P. Inverardi, J.-M. Jezequel, S. Malek, R. Mirandola, M. Mori, and G. Tamburrelli, "Perpetual assurances for self-adaptive systems," in *Software Engineering for Self-Adaptive Systems III. Assurances*, R. de Lemos, D. Garlan, C. Ghezzi, and H. Giese, Eds. Springer, 2017, pp. 31–63.

[35] M. Almaliki, F. Faniyi, R. Bahsoon, K. Phalp, and R. Ali, "Requirements-driven social adaptation: Expert survey," in *Requirements Engineering: Foundation for Software Quality - 20th International Working Conference, REFSQ*, 2014, pp. 72–87.

[36] W. Qian, X. Peng, J. Sun, Y. Yu, B. Nuseibeh, and W. Zhao, "O2O service composition with social collaboration," in *Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE*, 2017, pp. 451–461.

[37] M. Shaw, "What makes good research in software engineering?" *International Journal on Software Tools for Technology Transfer*, vol. 4, no. 1, pp. 1–7, 2002.

[38] M. S. Feather, S. Fickas, A. Finkelstein, and A. Van Lamsweerde, "Requirements and specification exemplars," *Automated Software Engineering*, vol. 4, no. 4, pp. 419–438, 1997.

[39] B. Vogel-Heuser, S. Feldmann, J. Folmer, J. Ladiges, A. Fay, S. Lity, M. Tichy, M. Kowal, I. Schaefer, C. Haubeck *et al.*, "Selected challenges of software evolution for automated production systems," in *13th International Conference on Industrial Informatics*. IEEE, 2015, pp. 314–321.

[40] J. Martinez, W. K. Assunção, and T. Ziadi, "Espla: A catalog of extractive spl adoption case studies," in *21st International Systems and Software Product Line Conference*. ACM, 2017, pp. 38–41.

[41] "Software Engineering for Self-Adaptive Systems Exemplars," https://www.hpi.uni-potsdam.de/giese/public/selfadapt/exemplars/, 2019.

[42] "*Cyber-Physiscal Systems Laboratory* at the Hasso Plattner Institute Potsdam," https://www.hpi.uni-potsdam.de/giese/public/cpslab, 2019.

[43] "*Virtual Experiences Laboratory (VXLab)* at the Royal Melbourne Institute of Technology," http://rmit.edu.au/vxlab, visited on 15 March 2019.

[44] J. O. Blech, M. Spichkova, I. D. Peake, and H. W. Schmidt, "Cyber-virtual systems - simulation, validation & visualization," in *Proc. of ENASE*, 2014, pp. 218–225.

[45] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (foda) feasibility study," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, Tech. Rep., 1990.

[46] J. Wuttke, Y. Brun, A. Gorla, and J. Ramaswamy, "Traffic routing for evaluating self-adaptation," in *Proc. of SEAMS*, 2012, pp. 27–32.

[47] A. Bennaceur, C. McCormick, J. García-Galán, C. Perera, A. Smith, A. Zisman, and B. Nuseibeh, "Feed me, feed me: an exemplar for engineering adaptive software," in *Proc. of SEAMS*, 2016, pp. 89–95.