Complexity Challenges in Development of Cyber-Physical Systems

Martin Törngren(✉) and Ulf Sellgren

KTH Royal Institute of Technology, 100 44 Stockholm, Sweden
martin@md.kth.se, ulfse@kth.se

Abstract. In embarking towards Cyber-Physical Systems (CPS) with unprecedented capabilities it becomes essential to improve our understanding of CPS complexity and how we can deal with it. We investigate facets of CPS complexity and the limitations of Collaborating Information Processing Systems (CIPS) in dealing with those facets. By CIPS we refer to teams of humans and computer-aided engineering systems that are used to develop CPS. Furthermore, we specifically analyze characteristic differences among software and physical parts within CPS. The analysis indicates that it will no longer be possible to rely only on architectures and skilled people, or process and model/tool centered approaches. The tight integration of heterogeneous physical, cyber, CPS components, aspects and systems, results in a situation with interfaces and interrelations everywhere, each requiring explicit consideration. The role of model-based and computer aided engineering will become even more essential, and design methodologies will need to deeply consider interwoven systems and software aspects, including the hidden costs of software.

Keywords: Cyber-Physical Systems · Complex systems · Complexity
Complexity management · Systems engineering · Software engineering

1 Introduction

The concept of Cyber-Physical Systems (CPS) was introduced 2006 in the US to represent the *Integration of computation, networking and physical processes where CPS range from minuscule (pace makers) to large-scale (e.g. national power-grid)*, (Cyphers 2013). Many definitions have followed, often emphasizing the large scale nature and CPS as *networks of physical and computational components*, (NIST 2017). The mainstream interpretation of the term "cyber" refers to the use of computers or computer networks, see e.g. (M-W 2017). However, the term actually originates from Norbert Wiener who coined cybernetics from the Greek "kybernetike", meaning "governance", referring to feedback systems. Today, both interpretations are relevant for CPS.

A key aspect of CPS is the potential for integrating information technologies, operational technologies in terms of embedded systems and control systems, and physical systems, to form new or improved functionalities. Common trends for CPS also include increasing levels of automation and integration across the design-operation

M. Lohstroh et al. (Eds.): Lee Festschrift, LNCS 10760, pp. 478–503, 2018.
https://doi.org/10.1007/978-3-319-95246-8_27

time continuum, so called DevOps. This positioning of CPS provides unprecedented opportunities for innovation, within and across existing domains. However, at the same time it is commonly understood that we are already stretching the limits with existing systems in terms of development of cost-efficient and trustworthy systems. Consider, for example, the roadmaps surveyed by the project Platforms4CPS (2017) and thrusts towards new systems and software engineering methods to deal with future CPS, (Jacobson and Lawson 2015). National Academies (2016) states the following: "*today's practice of CPS system design and implementation is often ad hoc, ... and unable to support the level of complexity, scalability, security, safety, interoperability, and flexible design and operation that will be required to meet future needs*".

Since future CPS are likely to be unprecedented in their complexity, it becomes essential to understand what characterizes such systems and how we can best deal with them. The line of argument of this paper is to investigate the nature of complexity of CPS through the following perspectives:

- Cyber-Physical Systems and environment, i.e. including other systems with which the CPS interacts, as well as the organizations developing the CPS.
- Limitations of Collaborating Information Processing Systems (CIPS) in dealing with complexity in developing CPS. We use the term CIPS to refer to humans and Computer Aided Engineering (CAE) systems that develop CPS.
- What current methodologies have to offer and what is lacking.
- Proposing ways forward to meet identified limitations and gaps.

In our work, we draw upon state of the art, discussions with industrial experts and our own experiences. Complexity issues relating to CPS is a daunting topic. During our work, we synthesized a CPS complexity view that brings together the above listed perspectives. We present this view in Sect. 2, since the synthesis serves well to introduce the topic, the concepts and the structure of the paper. The state of the art is assessed in Sect. 3, and Sect. 4 analyzes, in more detail, some of the identified facets of CPS complexity. Finally, in Sect. 5, we discuss our findings and draw conclusions.

Of the many types of CPS, see e.g. (Schätz et al. 2015) and (CPS 2016), we focus mainly on mechatronics and robotics applications, i.e. where the physical systems are synergistic configurations of mechanical, electrical and electronics technologies. We include humans as an integral part of developing CPS but do not, in detail, treat the role of humans as part of an operational CPS.

2 A View of CPS Complexity and Contributions

Our overall approach and view on CPS complexity is illustrated in Fig. 1. CPS are designed and realized by Collaborative Information Processing Systems (CIPS) – i.e. by human developers supported by CAE systems and available information and

knowledge. CPS operate within an environment[1] which may include other CPS, humans and other types of systems (nature made, social systems, etc., see Checkland 2000)[2].

In our view, we associate various characteristics – or *complexity facets* - with systems. As illustrated in Fig. 1, "system" may refer to a CPS itself, to the corresponding CIPS, as well as to the environment. The complexity facets have *consequences* (see bottom middle box in Fig. 1) for the abilities of humans and projects to deal with the CPS; that is to say, the facets will closely relate to the limitations of CIPS. To deal with these consequences, we thus need to provide adequate methods, theory and tools that address CPS complexity, aiming to bridge the gap with respect to limitations of CIPS; we refer to these as *bridging measures* (see middle box in Fig. 1).

Accordingly, a first contribution of the paper is to structure the various state of the art perspectives. A second contribution concerns a more detailed analysis of (i) relationships between various parts and aspects of a CPS and with its environment, and (ii) characteristic differences among software and physical systems in order to better understand barriers to their integration and some of the origins of complexity. As a final contribution, we identify key bridging measures. The paper structure as outlined in Sect. 1 relates closely to Fig. 1, with corresponding sections indicated.

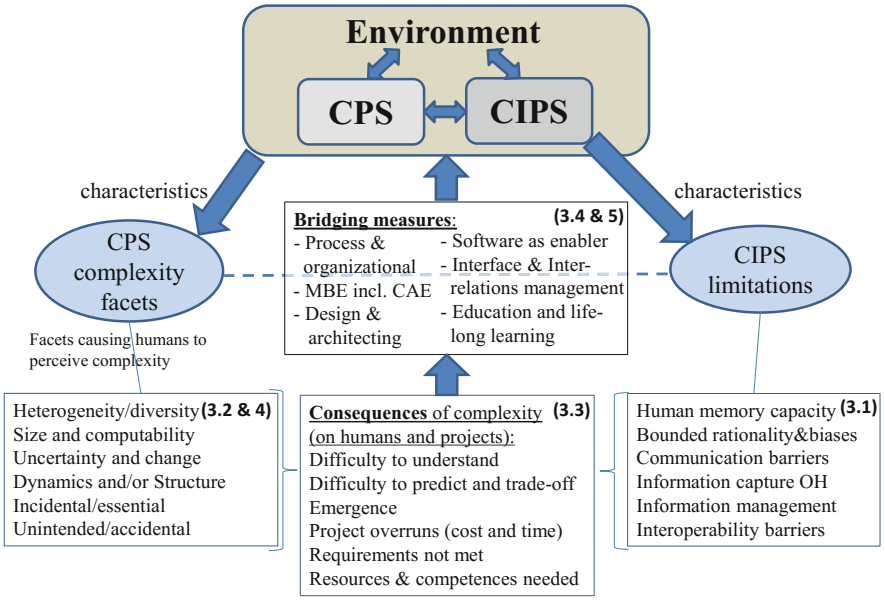

Fig. 1. A view of CPS complexity. The corresponding paper sections are shown in parenthesis.

[1] Other terms for "environment" include, for example, "wider system of interest", see e.g. Lawson (2015), or domain specific terms such as operational design domain, J3016 (2016).

[2] The context further includes other organizations and stakeholders, e.g. related to insurances, certification, legislation and standards; this context is only indirectly considered in the paper.

3 Engineering Practices Related to CPS Complexity

Related to Fig. 1, we provide perspectives from state of the art including CIPS limitations (Sect. 3.1), engineering views on complexity (Sect. 3.2), consequences of complexity (Sect. 3.3) and finally CPS engineering approaches to deal with complexity (Sect. 3.4).

3.1 CIPS Limitations in Dealing with Complexity

In this section, we study first humans and teams of humans, and then turn to CAE systems – all through the lens of limitations.

Limitations of CIPS Focusing on Humans: Examples of human limitations in dealing with information and (complex) systems can be found in studies in psychology and economy, see e.g. Simon (1996) and Kahneman (2012):

- *Capacity of the short term memory*; the short term memory is limited to holding and processing in the order of 7–10 "chunks" of information, where a chunk refers to one concept, which may be at different levels of abstraction and may refer to a more elaborate structure held in long term memory.
- *Capacity of the long term memory*; while having impressive storage capability with elaborate association mechanisms, the long term memory takes time to train and is not altogether reliable (see next point). As CPS requires deep knowledge in many areas, and as it takes substantial time for a person to become a deep domain expert, it follows that CPS development will have to involve many people. From our experience we note that it is very rare for a single person to be skilled in physics, logic and spatial concepts, all of which required for a holistic understanding of a CPS.
- *Bounded rationality and biases*; we humans are not as rational as we commonly think. There is, for example, a difference in what we experience and what we remember. Our brains, while mostly operating well, are prone to biases including overconfidence and a tendency to ignore or overemphasize the importance of small risks. Remembering is subject to neglect of duration and the "peak-end" rule, meaning that we give more weight to recent events. Furthermore, we are prone to search for, and to remember, pieces of information that confirm our current belief, which has a significant filtering and thus biasing effect.
- *Span of attention of the "slow" system*; the activation of what Kahneman refers to as the "slow system" of the brain, corresponds to what we could consider as "active thinking efforts". There is a resistance in activating the slow system since it requires considerable energy. Activating the slow system is beneficial when humans have to deal with novel considerations beyond their previous experience, where the "fast system" may not be able to come up with reasonable answers.

Organizations can overcome many of the limitations of a single brain by imposing processes involving, for example, reviews, checklists, detailed system analysis, and by supporting appropriate organizational cultures in the line of continuous improvement and constant quality control, (Kahneman 2012). These measures are closely in line with the best practices of systems engineering (INCOSE 2015).

However, additional challenges arise when dealing with advanced CPS, either in the form of very complex machines, such as a modern car, or in terms of a dynamically forming system of systems (SoS), for example in terms of a swarm of drones that together with infrastructure and other machines perform collaborative tasks.

A key aspect of these challenges refers to how to appropriately arrange communication among people, teams, organizations and CAE-systems, and how to organize them as well as the associated information. The design of an advanced CPS such as an aircraft or a car will require the collaboration of thousands of engineers. Communication among engineers in such settings is often seen as the key system development challenge, see e.g. (Andersson 2017). Organizing development will have to take into account the multitude of aspects and engineering phases of a CPS, while still facilitating proper interactions. Disciplinary experts are moreover schooled into various communities, theories and traditions, which introduces gaps in understanding among the experts, see e.g. Horváth et al. (2017) and Törngren et al. (2014).

For Cyber-Physical Systems of Systems (CPSoS), the challenge further relates to defining goals, policies and mechanisms for interactions among constituent CPS. As a key characteristic, a CPSoS involves CPS developed by multiple organizations where there is no clear responsibility for systems integration, see e.g. CPSoS agenda (2015). The intentions of the interactions within a CPSoS may be incompletely defined, misunderstood or interpreted differently by the involved organizations and experts.

Limitations of CIPS Focusing on CAE Systems: We now turn to limitations of CAE systems as part of CIPS. These include the following:

- *Dealing with tacit and implicit information, including context and meaning of concepts*. CAE systems require explicit formalization of information to be able to reason about CPS. For this time and resources have to be spent - when systems evolve, the information/models also have to be updated and kept consistent In the absence of fully collaborative CAE tools, development engineers use a large number of social communications tools, such as email and messaging. At present, there are very limited possibilities with current CAE tools to record communication interactions and histories into the CAE applications and then associate decision histories (and decisions) with the current design model (s) (Red et al. 2013).
- *Challenges in formalizing, managing and evolving the huge amounts of information and relationships required for CPS engineering*. Information management becomes difficult when considering different versions of components and assumptions and decisions made in developing artefacts such as models. Extra information is required to describe this context, further growing the amount of information. This is a significant challenge in multi-user development, see e.g. Red et al. (2013).
- *Limitations in interoperability and exchange among existing CAE systems*. CAE systems already hold a lot of useful information and models, albeit fragmented into different aspects or parts of a CPS, e.g. into software, electronics, and mechanics. Improved support for interoperability and exchange across CAE systems has the potential to drastically improve CPS management. While there are promising standards available, such as STEP and linked data, overall these have limited adoption so far, see e.g. Törngren et al. (2014), and El-khoury et al. (2016).

For the CAE system, the amount of information is, in itself, not a problem, but that information has to be made explicit. To support humans in development this extends to knowledge management, requiring access to appropriate up-to-date meta-data, clarifying the limitations and validity of the information. Such support will also require efficient interoperability among systems.

3.2 Engineering Views on Facets of Complexity

There are many interpretations and studies of "complexity" encompassing technical systems and humans, extending all the way to socio-political systems. There are also many propositions for metrics, definitions and facets of complexity. However, few metrics appear to be adopted into actual engineering practice. Definitions tend to focus on certain facets (see e.g. Sheard 2015). Frequently discussed facets of complexity include:

- *heterogeneity* of parts and interactions: CPS are strongly characterized by heterogeneity in several dimensions, with artefacts all the way from requirements, functions and technology to stakeholders. CPS represent hybrid, distributed, closed-loop as well as real-time systems, thus requiring developers to deal with a multitude of properties, behaviors and performance targets, see e.g. Derler et al. (2012) and Horváth (2017). As a result of their heterogeneity, CPS will typically be represented using multiple interdependent views, captured with different formalisms and tools, see e.g. Törngren et al. (2014).
- *size and computability related*: Large scale CPS will involve many things in terms of e.g. number of units, connectors, logical interactions, lines-of-code, requirements and stakeholders. Size related facets can also be seen to encompass the amount of information needed to describe an object (Shannon and Weaver 1949), the amount of resources needed to manufacture a product, (Suh 1990), or the computational complexity. The latter refers to the number of operations for solving an algorithm and how they relate to the size of the problem. Several CPS related design topics, such as assignment in space and time, belong to the class of NP-complete problems for which no polynomial time algorithms are known, see e.g. Blondel and Tsitsiklis (2000),
- *uncertainty and change*: Uncertainty can be used to refer to all kinds of unknowns in the context of system development. Uncertainty relates to complexity and risk by increasing the design space and potential for wrong decisions, and by complicating change management, Axelsson (2011). Typical examples include changing and conflicting requirements, unknown properties of technologies and impacts of design decisions. It can also refer to uncertainty of environment perception of a CPS see e.g. ESD (2003) and Sheard (2015),
- *dynamics or structure*: These complexity facets refer to either aspects of behavior that are difficult to predict, e.g. due to highly non-linear and coupled dynamics, or structural aspects such as dependencies among parts and properties. CPS typically represent tightly integrated and coupled systems where the change of one parameter in the design is likely to influence many other parameters. The behaviors and structures may also change dynamically such as in self-learning systems and in CPSoS.

A CPS typically also requires consideration of dynamics and structure at multiple levels or scales, e.g. from unit and subsystem to system level, and with different time horizons, see e.g. Sheard (2015) and Horváth (2017). Parallelism in terms of concurrent cyber and physical parts, and resource sharing in the computer systems further contribute to complexity, see e.g. Derler et al. (2012).

- *incidental vs. essential*: This facet refers to whether complexity arises from a particular way in which a system is designed (for example, due to the use of legacy components), as opposed to being inherent in the problem to be solved, (Brooks 1987). A key example of incidental complexity is that of improper design, or improper design assumptions that leave certain aspects of CPS design undefined, implying that side-effects may occur and/or that behaviors will emerge from the implementation rather than being designed. Examples of this include the lack of time abstractions and practices of hardware design, implying that timing behavior will emerge, see e.g. Lee (2009),
- *unintended and accidental behavior*: These behaviors refer to (known) side-effects or design faults, Qian and Gero (1996). For physical systems, unintended behavior represents a side-effect that may require additional sub-functions for dealing with (e.g. reducing) the undesired side-effect. The side effects are often of the same order of magnitude as the intended behavior and are typically caused by component interactions through their interfaces, e.g., friction-induced thermal effects between surfaces in contact (Whitney 1996). An accidental behavior is an unintended behavior that is caused by an accidental relationship or interaction between product features (e.g., a cable placed too close to a hot engine block). An accidental behavior is likely caused by a design error, Qian and Gero (1996). For cyber-systems, an example of a class of accidental behaviors is given by undesirable feature interactions not considered during design, see e.g. Broy (2010).
- *goals and socio-technical context.* This facet refers to the essential complexity of the goals in terms of their feasibility, see e.g. (Suh 1999; Maier 2007), and human/organizational aspects such as competition, conflicts, policies and management (Sheard 2015).

As noted by several authors, various facets of complexity can relate to different types of systems, including the CPS, the environment, and the organizations developing it (compare with Fig. 1), see e.g. Kaushik (2014) and Sheard (2015).

Some of the proposed facets can, at least in principle, be formulated in terms of absolute metrics (e.g. size related). Another type of metric is instead relative, for example in relation to what we try to accomplish or want to know, i.e. as a measure of the uncertainty of fulfilling the specified functional requirements (Suh 1999).

The evolution of CPS, towards more advanced functionality and operation in more open environments has implications for the complexity facets simply by providing "more of everything" including in terms of new or changed risks. As one key aspect, the increasing openness and large scale provide new attack surfaces that need attention to avoid increasing security risks. It will no longer be possible to a priori foresee all scenarios and what might go wrong so dynamic risk management may be necessary (see e.g. Boyes 2013). However, adding more protection mechanisms may further increase system complexity. Uncertainty needs to be considered, for example in

sensing, see e.g. Sadigh and Kapoor 2016. While the introduction of AI in terms of machine learning into CPS provides new capabilities, it also increases complexity. The robustness aspects of machine learning systems are currently not well understood, with implications for robustness and safety, see, for example, Wagner and Koopman (2015).

The design space of CPS illustrates several of the complexity facets (including e.g. heterogeneity, size and computability, and uncertainty), with a potentially very large number of design choices and dependencies among desired properties related to design decisions, which, in turn, requires trade-offs to be made (see e.g. Maier and Rechtin 2002). In early development stages, designers have considerable freedom with respect to design decisions, but no full insight into the implications of those decisions. Later in the development process, when they have acquired more knowledge, by experimenting with various models and physical prototypes, they will have less degrees of freedom, because of the decisions made upstream in the process. This dilemma is sometimes referred to as "the cone of uncertainty", e.g. (McConnell 1997) as represented in the right portion of Fig. 2. The cone of uncertainty also emphasizes the view that complexity, as a measure of uncertainty (Suh 1999), is reduced as we learn when we proceed with the development.

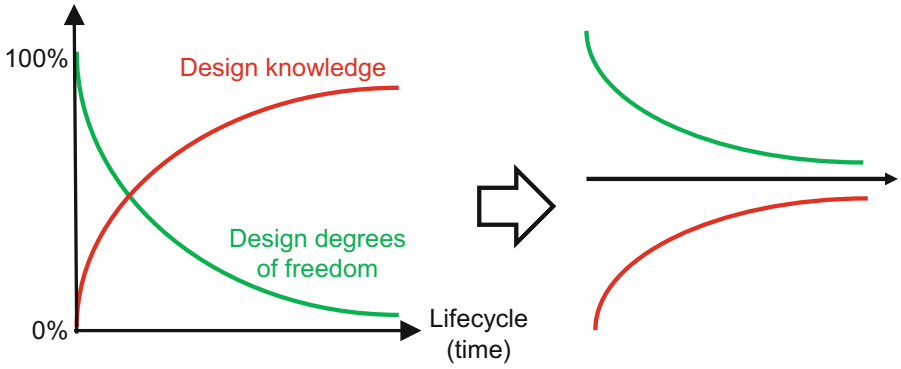

Fig. 2. The cone of uncertainty

We conclude that the interaction and co-existence of the cyber- and physical parts, as well as their development context, give rise to all of the described facets of complexity. This will be further highlighted in Sect. 4.

3.3 Consequences of Complexity

Complexity can be viewed in terms of what Sillitto (2009) referred to as "objective complexity", referring to technical or engineering characteristics, or in terms of "subjective complexity", relating to how humans perceive the systems, see e.g. Sheard (2015). An example of the former would be metrics of the inherent problem size and algorithmic complexity involved in optimizing a CPS. Perceived difficulty of understanding the behavior of a CPS would be an example of the latter. Sheard (2015) makes

the interesting observation that while objective complexity is always growing (we build more and more sophisticated systems), the subjective complexity may, in fact, reduce over time once new systems become accepted and better understood.

As depicted in Fig. 1, in this paper we take the approach to try to make these ends meet by contrasting limitations of CIPS (including humans) with various characterizations – facets - of complexity. We believe this is one fruitful way forward since the CIPS will have to deal with the CPS.

Recalling Sect. 3.1, it is not strange that even a moderate CPS poses challenges for humans, who may find even the single facets of CPS complexity difficult to deal with. Combining multiple facets of complexity implies that it becomes non-trivial to predict the behavior of the system, and to understand the impact when making changes in the system. This also implies that trade-offs become more challenging and potentially more subjective.

The concept of emergence is often used when discussing complexity; it is a term that has been given several interpretations. We here use the following one, closely related to the difficulty in predicting system behavior: "The whole is more than the sum of the parts, in the sense that given the properties of the parts and the laws of their interactions, it is not a trivial matter to infer the properties of the whole", (Simon 1996). Emergence stems from difficulties in understanding the effects of interactions among parts and can have positive or negative consequences (recall unintended and accidental behaviors, described in Sect. 3.2).

It is interesting to research the impact of complexity on projects. Sheard (2015) investigated how a number of complexity-related variables (or metrics) contributed to project cost overrun, project schedule delay, and system performance shortfalls. 39 variables in 75 development projects were investigated through a retrospective survey with senior system engineers and project managers. The following three complexity variables were found to correlate positively with problematic outcomes in all three aspects (cost, schedule and performance): (i) number of hard-to-meet, and frequently also conflicting, requirements, (ii) degree of cognitive fog[3], and (iii) stability of stakeholder relationships.

3.4 How is Complexity Dealt with in CPS Engineering

A multitude of approaches, methods and tools have been developed over the years to deal with CPS complexity. In this survey we focus on the following:

(i) *process*, (ii) *model-based and computer aided engineering*, (iii) *design and architecting*, and (iv) *people/organizational*. These approaches are complementary and partly overlapping. There is no silver bullet for dealing with complexity, as phrased by Brooks (1987). Many of them involve ways to divide and conquer a system (in terms of the CPS, the development teams, models etc.) into separate parts to facilitate their management. Despite the importance of so-called front-loaded development, where design decisions and means to improve the management of uncertainty and risk are key

[3] With (ii), the question posed was as follows: "The project frequently found itself in a fog of conflicting data and cognitive overload - Do you agree with this statement?".

elements, such practices are still often weak. Integration is generally identified as a (time and cost consuming) challenge, see e.g. INCOSE (2015). Simmons (2005) reports that only a very small amount of the total development efforts is spent on systems architecting, despite the crucial decisions taken in that phase.

Process Approaches: Systems engineering methodologies describe a number of recommended processes, from technical to management, see e.g. INCOSE (2015). Figure 3 illustrates an elaborated V-model for mechatronic systems – focusing on the technical process, where development is divided into stages and into engineering disciplines, VDI 2206 (2004). The decomposition approach is accompanied by providing guidance for risk management (project and product risk), and for step-wise integration of the decomposed entities. The conventional use of rapid prototyping, code generation and various X-in-the loop simulation schemes (e.g. software- and hardware-in the loop) provide examples of this. Software engineering methodologies emphasize agile approaches involving close collaboration within teams, frequent releases and close interactions with stakeholders, see e.g. INCOSE (2015), which helps to reduce development uncertainty and risk. We note that agility is one means to deal with uncertainty (one identified complexity facet) and risk. Nevertheless there are challenges in reconciling agile approaches with safety critical systems development, see e.g. Axelsson et al. (2015). Most disciplinary CPS development today involves frequent iterations and developments in smaller steps, whilst also requiring explicit considerations of the synchronization between software and hardware parts throughout the development and production phases, see e.g. Jacobson and Lawson (2015).

Model-Based and Computer Aided Engineering Approaches (MBE): With MBE we refer to approaches that make systematic use of abstractions and of computer engineering tools, i.e. including CAE, to deal with CPS complexity. Abstractions provide the means to focus work on particular aspects, while neglecting other aspects that have less influence on the issues at hand. To deal with the "Cone of uncertainty" (recall Fig. 2), models and their analysis e.g. through simulation offer ways to increase problem understanding, explore uncertainty and the solution space. Synthesis based on models help to improve efficiency of development by automating certain design steps, thereby removing certain sources of faults, see e.g. Törngren et al. (2008).

Many CPS constitute closed loop systems, implying that MBE approaches are necessary for efficiency; for example, without a model of a controlled system, control development cannot start until the physical system is developed. The closed-loop aspect has also led to a widely accepted use of models in verification; the system behavior arises through the closed loop interactions between the cyber and physical parts. A further key aspect of MBE is that of model verification and validation, ensuring that models are as simple as possible yet adequate for the intended purpose.

The success and increasing use of models has led to a need to emphasize their composition and management. Models become systems in their own right, with assumptions, interfaces, versions and variants relying on modeling environments. Model management is a research area with a surprisingly large number of still open challenges. Efforts in this area stem from a variety of directions including product-life cycle management (mechanical engineering) and application life-cycle management (software engineering), see e.g. Törngren et al. (2008), towards CPS life-cycle

management. Correct composition and usage of different types of models can be supported by the use of contracts and explicit dependency models, (see e.g. Westman 2016; Qamar 2013) and references therein, and through uncertainty management, (see e.g. Mohan et al. 2017). Correct composition of simulation models reflecting different types of behaviors and concurrency is essential for CPS (see e.g. Derler et al. 2012). Divide and conquer approaches are also applied to models, through multi-view models and multi-view frameworks. An example of this the CPS architectural framework initiated by NIST, which provides common viewpoints such as functions and interfaces, as well CPS specific aspects such as trustworthiness and timing (NIST 2017).

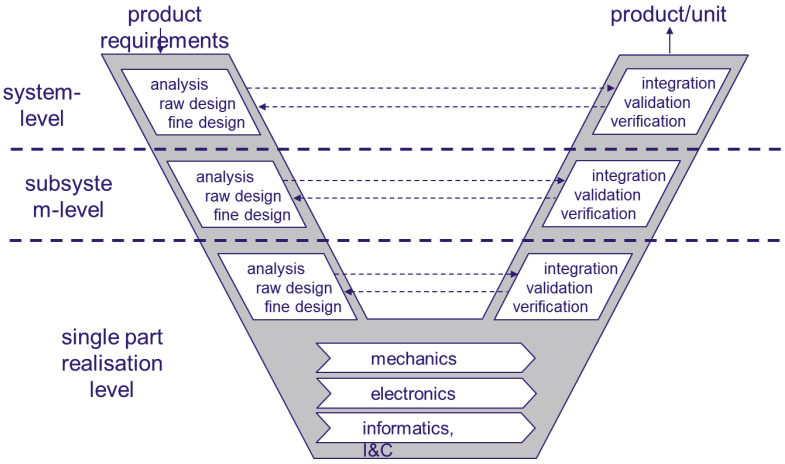

Fig. 3. Design methodology for mechatronic systems development, VDI 2206 (2004).

Design and Architecting Approaches: In this category we include principles, methods, and techniques that aim to reduce the incidental complexity or better manage the inherent complexity of a CPS. Examples of such approaches include (i) the use of deterministic execution platforms to reduce side-effects, facilitating understanding, integration and verification, (Kopetz 2011), and (ii) modularization techniques that use metrics for establishing "low coupling" and well defined interactions at interfaces between modules, see e.g. Börjesson (2014). Lee (2016) recommends the use of deterministic design models as far as possible, since it facilitates the design of complex systems by enabling definitive analysis. It should be noted that deterministic models can efficiently be utilized in probabilistic analyses, e.g. based on Monte-Carlo simulations, to provide knowledge on the effects from known or speculated variations in CPS design, environment and/or operation. An important and complementary approach to reduce incidental complexity is, of course, to reduce some of the essential complexity if that is possible, by relaxing some of the requirements of a system.

Because of the large number of involved stakeholders (including multiple organizations) it is important to realize that it will rarely be possible to optimize a large scale

CPS. Rather it is important to try to find solutions which are satisfactory to the involved stakeholders, (Simon 1996) and (Kahneman 2012).

People and Organizational Approaches: Skills of people and organizational designs are clearly imperative in dealing with CPS complexity. Organizational integration mechanisms including organizational structure, work procedures, training, social systems, and CAE have been shown to be important for improving organizational performance, (Adamsson 2007). Coordination among CAE systems is consequently also essential. Referring to Fig. 3, an organizational structure - such as dividing into mechanical, electronics, software, etc. - will also often imply a division of information and CAE systems along the same structure, leading to potential problems in managing interactions among teams as well as between CAE systems, see e.g. Malvius (2009).

One further important aspect of organizational design is that of "intelligent information filtering", providing people with adequate information that suits their purposes as part of the development whilst avoiding information overflow, (Simon 1996).

4 Analysis of CPS Complexity Facets

In this section we further investigate specific CPS facets of complexity that, to our understanding, are important, but have not received the attention they deserve. These facets include interrelations related to CPS (Sect. 4.1) and characteristic differences among software and physical systems (Sect. 4.2).

4.1 CPS Component Interrelations and Their Implications

We first turn to a CPS component perspective to analyze interrelations. A first relevant question to ask is: what constitutes a CPS component? CPS exist in the small and in the large. Compare, for example, a modern milling machine within a production cell with a manufacturing system that incorporates multiple production cells and their coordination, forming a distributed computer control system. In a CPS that involves humans, e.g. as operators, humans also become "components" within the CPS. The assignment and division of responsibilities among humans and other components is important (although out of the scope of this paper).

Figure 4 illustrates two CPS components of a mechatronics machine: the components are interconnected physically (illustrated in the middle of the figure), and through a communication network (illustrated through the horizontal line labelled "Communication network", connecting the communication subsystem of each component). This illustration would be relevant for vehicles (e.g. cars and airplanes) and production machines, where each component (e.g. brake, engine, transmission, etc.) incorporates mechanical parts and computing. As apparent from Fig. 4, there will be many interactions between the various parts, including between the cyber and physical parts. The direct interfaces between the computer system and the mechanical parts, through sensors and actuators, are of course crucial. However, beyond this, the mechatronic components interact physically with each-other and with the mechanical frame on which they are mounted. The components further interact through information

exchange and through the energy subsystem, and they will also have interactions with the environment, e.g. through heat, noise and electromagnetic radiation.

This increasing connectivity enables direct collaboration among machines with external resources such as edge or cloud computing resources. Considering a larger scale CPS, its subsystems and components may thus also include machine external computing and communication resources. We note that such computing and communication resources, while normally considered to be part of the cyber-side of a CPS, indeed also constitute cyber-physical systems in their own right, since they are composed of software, analog and digital electronics, power supplies, cooling and mechanical parts. Design of such CPS will necessarily have to consider and thoroughly manage interactions and integration among these cyber and physical parts.

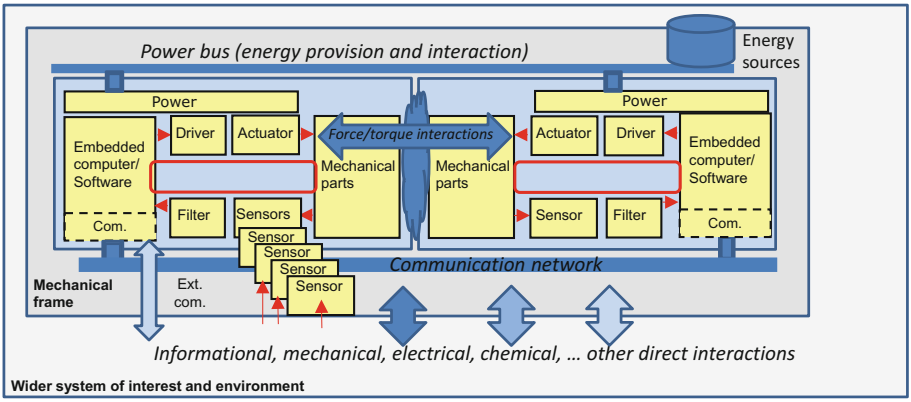

Fig. 4. Illustration of (two) mechatronic components and their various interactions with other components and the environment. Each component may have internal and external sensors. Note that "driver" refers to electronics for the actuators.

Now consider again mechatronic components and the example of interactions between the major components of a car. Figure 5 provides a Design Structure Matrix (DSM) representation of a pre-1970 car (left), vs. an early 21st century car (right).

	Susp	Brake	Steer	Wheel	Diff	Trans	Clutch	Eng	Driver
Susp				X					X
Brake				X					X
Steer				X					X
Wheel	X	X	X		X				
Diff				X		X			
Trans					X		X		
Clutch						X		X	X
Eng							X		
Driver	X	X	X				X		

	Susp	Brake	Steer	Wheel	Diff	Trans	Clutch	Eng	Driver
Susp		P	P	X+P	P	P	P	P	X+P
Brake	P		P	X+P	P	P	P	P	X+P
Steer	P	P		X+P	P	P	P	P	X+P
Wheel	X	X	X+P		X				
Diff	P	P	P	X+P		X+P	P	P	
Trans	P	P	P	P	X+P		X+P	P	P
Clutch		P	P		P	X+P		X+P	P
Eng	P	P	P	P	P	P	X+P		P
Driver	X+P	X+P	X+P		P	P	X+P	P	

Fig. 5. DSM representation of a pre-1970 (left) vs. a modern car (right), illustrated through two types of interactions between major car components: X – physical connection and force interaction; P – Programmable relations.

A component-DSM may be used to illustrate different relationships (such as energy, mass, information flow and spatial interactions) between components in a system, e.g. (Steward 1981; Eppinger and Browning 2012). In the left part of Fig. 5, this is illustrated with mainly mechanical interactions, without distinction of the type(s). We note that the DSM-model displays symmetry that is typical for a mechanical system with bidirectional Newtonian interactions. In a modern vehicle, all major components will also have integrated embedded systems, and there is thus an opportunity for information interaction and additional collaboration between components. As an example of this, consider the connections between the steering and the wheels of the car. Apart from a traditional mechanical connection, the steering of a modern car may today also be controlled by a "vehicle stability controller", able to apply individual wheel braking in order to deal with unintended car yaw. This explains the "X" connecting "Steer" with "Brake" in Fig. 5. The control system is able to act much faster than a human driver, and can thus avoid many accidents (subject to key constraints such as the condition of the tires and the road surface condition).

The DSM to the right in Fig. 5 illustrates the large potential in introducing novel functionalities to improve performance. These interactions also clearly illustrate the growing complexity, in terms of heterogeneous components and multiple interactions, with difficulty in predicting and verifying the final behavior. This setting also clearly leads to organizational challenges in setting up clear responsibilities to deal with the interactions/relationships among functions, components, properties, teams and activities. Considering versions of software and variants (e.g. in terms of features in a car due to customer choices or market requirements) further complicates the scene. The car example concerns a tightly coupled dynamical system that strongly incentivizes adding additional cyber-connections (thus increasing complexity). We believe that such interactions will similarly be driven also for less tightly coupled systems as cost-efficiency improves and opportunities for new services arise.

For large scale CPS, e.g. in terms of cloud connected and collaborating vehicles forming a CPSoS, we can clearly envision a DSM representation that expands with more components and interactions.

Beyond direct interactions as so far discussed - for example, between software components, and between sensors and the vehicle environment - a CPS will also importantly feature several types of indirect relations, see Fig. 6.

In particular, most components will be inter-related through assumptions made during design. For example, the software and the algorithms it embodies (e.g. for control and signal processing) are developed based on assumptions of the properties of the mechanical system. Similarly, the software components may incorporate assumptions about the electronics hardware, and the mechanical components may have assumptions w.r.t. the electronics hardware (e.g. size and weight). We note that the case of assumptions is generally valid for both cyber and physical parts, and that these assumptions create dependencies that need to be understood and managed.

To conclude this analysis, we would like to tie the discussion back to Fig. 1, which at the top illustrates relationships between the CPS, the environment and the CIPS. As stated by Simon (1996), a system will be "molded" by purposes related to its environment. In other words, it can be expected that a CPS will have an essential complexity that somehow corresponds to the complexity of the environment with which it interacts.

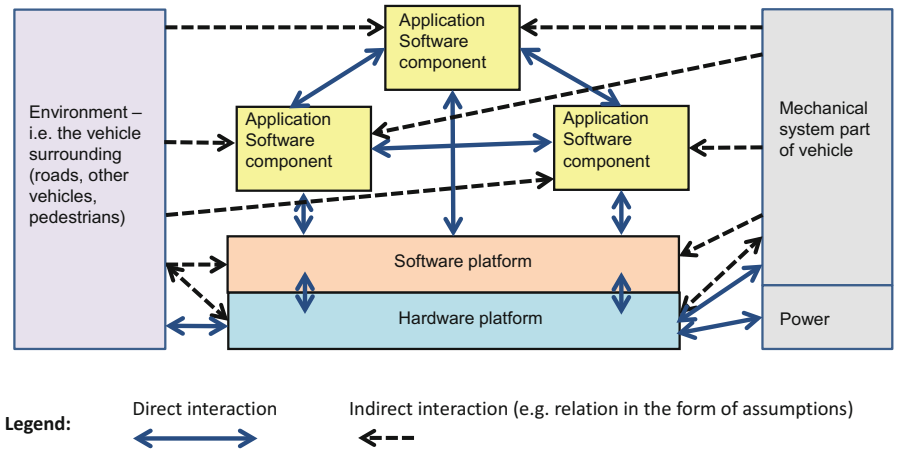

Fig. 6. *Direct* and *indirect* relations among CPS parts. Direct information and force interactions take place between parts. Indirect interactions refer to assumptions made in designing a part (the direction of the arrow indicates for which part the assumptions were made).

Consider, for example, the design of a highly automated vehicle (our CPS of concern). To develop this CPS we have to consider its operational context in terms of a multitude of relevant driving scenarios including static and dynamic objects that we may encounter; this emphasizes a number of complexity facets such as heterogeneity, size, dynamics and uncertainty. The complexity of the environment and the required functionality thus requires a lot from the CPS, driving its complexity. Similarly, in designing an organization to develop CPS, the complexity of the CPS will require a lot from the CIPS, thus driving their complexity. The relationships as shown in Fig. 1 are bidirectional in the sense that the systems (CPS, environment, CIPS) are influencing each other.

Finally, for organizations to be effective, investigations have indicated that the DSM product structure should be closely mirrored by corresponding organizational DSMs (i.e. relationships among teams) and processes (relationships among activities/steps)[4], Eppinger and Salminen (2001). In times of technology change it becomes especially important to keep these various "architectures" in sync. CPS embraces a paradigm shift with drastically new functionalities and components continuously being added to the technical architectures, which themselves must evolve. Thus the organizations and the processes need a corresponding evolution.

[4] Such DSMs are often referred to as team-based and activity-based, e.g. (Eppinger and Browning 2012).

4.2 Analysis of Distinguishing Characteristics: Physical vs. Software Systems

Dealing with the physical vs. cyber side represents specific challenges, since each side comes with very different traditions and expected properties, for example, from very fast turn-around, open and security aware systems to safety-critical real-time closed systems. We will here focus on software within the cyber-part, since we believe this pinpoints the essential differences. Table 1 summarizes distinguishing characteristics among physical and software systems, elaborated as follows.

Phenomena and Dependencies: A key concern of physical systems is their multitude of aspects encompassing structure, material properties and various types of dynamics in terms of e.g. stresses, heat, motion, vibrations and wear. As discussed in Sect. 3, side effects are often of the same order of magnitude as the intended behavior. Wear, tear and imperfect production imply that parameters will differ from nominal specifications (albeit within tolerances), i.e. they have distributions, and that they will change over time. The natural variation of the physical parameters, such as dimensions and material properties, make the behavior - and thus performance - probabilistic, requiring probabilistic analyses.

In contrast to physical systems, there is an apparent ease with which functions are realized in software systems. This ease relies upon abstraction hierarchies, a multitude of tools and existing software components that enable a direct path from software programs to their execution by microprocessors. Software is an abstraction notion that provides powerful and flexible constructs for describing information, logic and algorithms, without direct physical constraints. This enables us to build systems of unprecedented size, to the point where an incredible state-space complexity is created. This evolution has led to modern cars being fitted with tens of millions of lines of code and large parameter sets, see e.g. Broy et al. (2007).

Table 1. Contrasting characteristics of physical vs. software systems.

	Physical systems	Software
Phenomena & dependencies	Multiple coupled physical phenomena (materials, wear, fatigue, heat, …) Local direct effect	State space size; bugs; connectivity; variability Local and global direct effects
Dev. Time & iterations	Long (manufacturing)/few iterations	Short/long; large amount of iterations
Abstractions, synthesis, and platforms	Approximations; continuous time and value; No single platform - multiple realization technologies; Behavioral model sim.; Geometry based synthesis (CAD/CAM/); Form as a component or structure property	Digital abstractions; discrete time and value/strong platform foundations; Property preserving model transformations (code synthesis)
Extra-functional (EF) properties including cost	Trade-offs among EF properties; Established cost models	Dependencies create additional relations between EF properties; Difficult to estimate life-cycle cost

A direct consequence of the abstract nature of software is that it only has design faults. Physical systems are, on the other hand, characterized by design faults, random hardware failures and wear-out faults, i.e. faults caused by frequent operations/usage. A consequence of the inherent complexity of software systems, in particular at large scale (see e.g. Brooks 1987), is that design faults are much more predominant in the cyber-side. 10 bugs per 1000 lines of code is commonly estimated for commercial software, with in the order of one bug per 1000 lines for safety critical code, McDermid and Kelly (2006)[5].

Further distinguishing characteristics w.r.t. phenomena and dependencies are that physical systems have strong interactions locally, with weaker remote effects (Simon 1996). These effects can, in many cases, be seen as piece-wise linear. In software and digital systems, potentially any bit-flip or bug may break the system. Cyber-systems are, in this sense, highly non-linear, (Henzinger and Sifakis 2006). Since software systems (executing on hardware) in principle can be provided with very high connectivity, any change or fault or just nominal communication has the potential to have a large impact globally, unless the design explicitly takes this into account. The resulting systems may then come to violate the natural "architecture of complexity" (Simon 1996). Such systems - without barriers, where everything is interrelated - are likely to be brittle and unmanageable.

Development Time and Iterations: In a CPS project of any size, there is a substantial difference in the number of iterations used and time duration for the development of the software, electronics vs. physical parts. As an example, the duration of a project designing a new industrial computer could be in the order of 1-1.5 years. During this period, 2-3 mechanical prototypes, 3 iterations of electronics, and 100 iterations of software might be provided. For mechanical products, design and manufacturing take a considerable amount of time and effort. For electronics, dealing with heat, isolation, ruggedness etc. requires extra consideration and time.

While software does not need the same type of production effort as physical systems, it nevertheless heavily relies on a host of previous developments – a "software infrastructure" - including tools, operating systems, middleware, libraries, and existing application components. This infrastructure is growing over time, and increasingly includes, for example, capabilities to upgrade software and to gather data from running systems. The time to develop software will therefore be strongly dependent on the availability of a proper software infrastructure. There is a tendency that too little emphasis is placed on the software platform (Ericson 2017). According to industrial developers, software development never ends, and is never ready when the product is delivered. The software complexity also gives rise to concerns for effective verification and validation.

Abstractions, Synthesis and Platforms: Digital hardware platforms enable abstractions (programs, code) to be converted into executable/interpretable code. This relies on abstractions of services at different levels, e.g. processor instruction sets and programming instructions to higher-level services that define the basis for even more services. These abstractions of services are often referred to as platforms for digital

[5] The number of bugs is only used here to illustrate the complexity; not all bugs are equally important.

systems. This notion is captured by so called Platform-based design (PBD), stemming from the field of Electronics Design Automation. Defining and constraining the set of platforms has the effect of reducing the design space, increasing reuse, and speeding up development, while allowing focus on application development and its mapping (and tailoring) of the platform, see e.g. Sangiovanni-Vincentelli (2002).

The powerful foundation of abstractions and digital platforms have proven very successful for general computing but become a problem for cyber-physical systems, since they do not cover physical effects such as timing and energy consumption, see e.g. Henzinger and Sifakis (2006), and Lee (2009).

For physical systems, we first note that the term platform is used differently, to refer to the "common necessary modules" of a product, Blackenfelt (2001). Compared to software, physical systems have no corresponding general realization platform. Instead, there are multiple candidate technologies, for example for actuation in terms of electrical, hydraulic, or combustion engine technologies. Functional and behavioral descriptions provide goals that will be approximated by the realization technology (good enough) but also leading to side effects as discussed previously.

A second somewhat subtle difference between software and physical systems refers to abstractions and their relationship to synthesis. For software systems, behavior abstractions are synthesized (refined) into executable code. Behavioral models are also common to support physical systems design and analysis, e.g. used for evaluation by simulation. However, synthesis in the form of manufacturing relies on geometry rather than behavior models. Synthesis is thus based on geometrical descriptions (CAD) that can be transferred to computer aided manufacturing systems. It is **not** a given that the geometrical representation adequately represents behavioral models. As an additional aspect of relevance, the geometrical form is an important attribute of physical systems (and thus of CPS). The form may correspond to a structural property or be realized through specific physical components. New manufacturing paradigms, such as additive manufacturing, expand design freedom by removing manufacturing constraints on shape, material combination and product structure imposed by traditional machining operations.

A third difference refers to the view on time and values, with continuous abstractions dominating at the macro-level in the physical world, and with discrete representations in the software (and digital world), leading to quantization and discretization concerns when integrated into CPS.

Extra-Functional (EF) Properties Including Cost: Many physical related EF properties, such as reliability and safety, only become concrete for software when considered in the context of processing hardware together with software. Alternatively properties such as reliability are considered as controversial when applied to software only. The nature of software leads to special considerations for flexibility-related EF properties such as upgradeability and maintainability.

A specific concern for software systems is that they will - for cost and interaction reasons - be sharing various resources such as computing and communication elements as well as data and algorithms. This has the implication that many extra-functional properties will be highly dependent on shared elements and design parameters (e.g. the speed of a network and policies of a server). Unless care is taken in design, the sharing may contribute to complexity by introducing design faults, such as undesirable feature interactions mentioned in Sect. 3.1.

Cost models appear to constitute an industrial challenge especially for software. While hardware costs are relatively well understood, software costs are more difficult to model and predict due to the described characteristics, e.g. accounting for the costs of the software platform, verification, and maintenance. A typical implication of this lack of awareness is that emphasis is often placed on reducing hardware costs, while software costs are disregarded. An example of this would be the introduction of two hardware platforms (of different costs), suited to different customer segments, even though this causes the software complexity (and therefore cost) to increase in order to deal with two variants. Another example relates back to resource sharing, where the drive to reduce hardware costs results in additional engineering effort to ensure that algorithms, computing platforms and available memory together meet the requirements e.g.in terms of accuracy, speed and predictability.

5 Discussion and Concluding Summary of Bridging Measures

5.1 Discussion

Reconsidering Fig. 1, the development of CPS has to consider its physical and cyber parts, the CIPS, and the environment. Facets of complexity appear in each of these "systems" and can moreover be considered for different aspects of these systems, including their behavior, structure, requirements, and relations among those and with external system aspects.

Figure 7 provides a corresponding elaboration of complexity facets applied to different types of systems and aspects of those systems. Fig 7 draws inspiration from Sheard (2015) in the distinction between the top and lower level. The complexity facets (right bottom box in Fig. 7) are the results of the analysis in this paper. The system aspects (left bottom box in Fig. 7) roughly correspond to key systems engineering development steps, see e.g. Oliver et al. (1996).

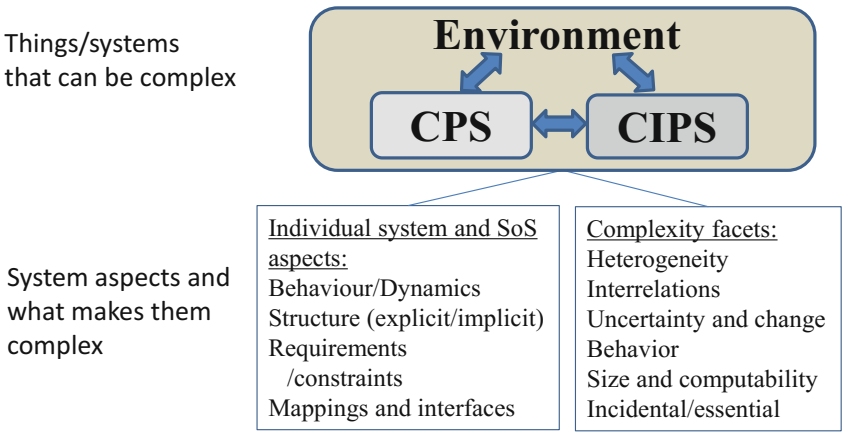

Fig. 7. Complexity facets applied to different types of systems and aspects of those systems.

Our review of the state of the art and our analysis reveals that all aspects of complexity discussed in this paper are relevant for developing future CPS. The analysis, however, reveals that the facets identified in Fig. 7 are of particular importance.

In particular, the tight integration of physical and software parts, and CPS components, results in a situation with interfaces and interrelations everywhere (compare with Figs. 4, 5 and 6). The properties of the complete product appear as a result of the component, software and physical system properties and their interactions. Intricate relationships between components will contribute to a number of properties such as functionality, performance, safety, security, flexibility and interoperability. Changes in some component properties or interrelations may affect multiple properties, in essence leading to tensions that will require these interrelations to be understood and that appropriate trade-offs are made. It thus becomes central to manage both explicit and implicit interrelations, including uncertainty in information. This is of relevance for all the types of systems depicted in Fig. 7, i.e. the CPS, the environment and CIPS.

Development of CPS, moreover, has to face the combined consideration of physical and software facets of complexity, including those described in Table 1. One important aspect of this is the need to enhance a mutual understanding across cyber- and physical (related) disciplines, posing an educational challenge.

The significantly increasing system complexity for CPS, compared to traditional systems, has the effect to increase the uncertainty that remains when a new system is launched to the market as well as the amount of information (models, data, etc.) required to describe the CPS. Recalling the "cone of uncertainty" in Fig. 2, this corresponds to a widening of the cone – see Fig. 8; the more complex a system is, the more uncertainties will remain even after the system has been deployed.

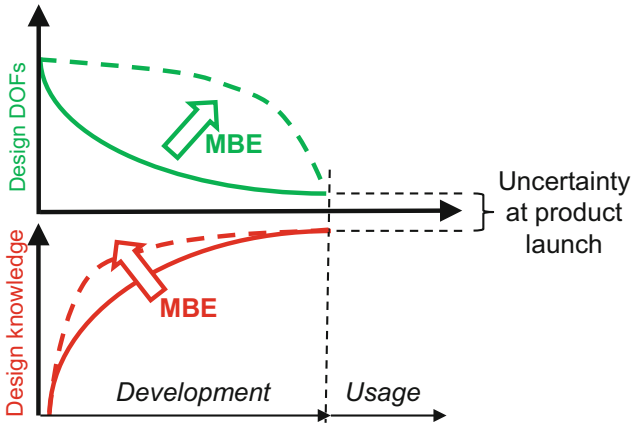

Fig. 8. MBE may enable early knowledge capture and deferred decisions

Such a situation may require development to be extended to the usage life-cycle phase, with diagnosis, condition monitoring and proper management of service and maintenance information to guide a process aimed at refining the CPS with continuous uncertainty reduction.

CPS development provides more options and thus more degrees of freedom (DOF's) in design compared to traditional systems, consequently requiring more knowledge to manage uncertainty and risks. The role of model-based engineering (MBE) is indicated in Fig. 8. MBE has a potential to create design knowledge at a significantly faster pace than design by physical prototyping, and also enables decisions to be delayed, i.e. keeping the design degrees of freedom for a longer time, see e.g. Sellgren 1999. There is consequently a very strong need to systematically implement methodologies that are model-based and where a reduction of the knowledge gap is driven by model-based analysis.

5.2 Bridging Measures Concluded

In the state of the art (Sect. 3.4), we described four approaches for dealing with CPS complexity. We believe that these approaches are very valid but will not be enough for future CPS. Trends towards connectivity, new services, automation, smartness, etc. imply that we are embarking towards both CPS and CPSoS of unprecedented complexity. Such systems will, in turn, be developed by large-scale CIPS. These trends unfortunately place further stress on the described limitations of CIPS. It thus becomes even more important to emphasize bridging measures.

The analysis indicates that it will no longer be possible to rely only on a subset of the approaches covered in Sect. 3.4. That is, approaches that we have encountered in industry for dealing with complexity, for example, placing specific emphasis on architectures and skilled people, or process and model/tool centered approaches, will most likely not suffice for the CPS of tomorrow. Instead, there will be a need for a broader set of tools – "bridging measures" to deal with future complexity. In the following we briefly summarize such bridging measures. These include the four approaches from Sect. 3.4, here grouped into three "reinforced" measures. To these we add measures to deal with software, including its hidden costs and as part of integrated design methodologies for CPS, interrelation management (drawing upon Sect. 4.1), and finally, education, ending up with the following six measures:

- *Processes and organizations for CPS.* Processes and organizations for CPS need to be able to explicitly address synchronization and integration among the diverse aspects and parts of a CPS, and consider integrated life-cycle engineering. The difference in speed of development of software and hardware needs explicit attention (synchronized processes, version and variant management, agile vs. safety practices), supported by architectures, and verification and validation methods. Key aspects for the successful development of CPS include insightful leadership and the use of integration mechanisms among teams for large scale CIPS.
- *MBE including CAE systems and frameworks for data management as design assistants*. Humans and organizations will need much better support for dealing with future CPS. Considering large scale CIPS (and CPS), means to support efficient and effective communication among people/teams will become even more important. Examples of areas with strong potential for dealing with the consequences of complexity (recall Fig. 1) include visualization, augmented/virtual reality, traceability and change management (e.g. managing interrelations), data

analytics, automation, and improved support for large-scale concurrent engineering. Advances in CAE capabilities with respect to semantic and contextual understanding and in dealing with large amounts of data will be necessary. Progress in AI is likely to provide entirely new capabilities to deal with many of those issues, including data analytics, model synthesis, and in providing decision support. Considering DevOps, CAE systems and underlying theories, concepts and strategies need to be developed for information and knowledge management that encompasses the entire life-cycle. A better understanding is also needed for how to balance static analysis, simulation, and physical tests, and how they can complement each-other.

- *Design and architecting.* New architectures and architectural representations are needed to support safety, security and availability, while managing evolution including software upgrades. Principles, interaction protocols, and architectures are also needed to support scalability, robustness and avoidance of side effects among interacting CPS parts of a CPSoS. Further, new methods and models are needed that explicitly manage relationships between extra-functional properties, incorporate uncertainty and concepts of dynamic risk management, attempting to mitigate risk even in the face of the unknown.
- *Software as enabler:* Software comes along with hidden costs and relies on extensive software assets that deserve attention because of their critical impact on end system properties. Better insights and cost models are needed to improve awareness. Software systems form an essential and growing part of CPS. Development methodologies need to incorporate core aspects of both systems and software engineering. The software communities need to embrace and explicitly consider the various direct and indirect physical effects of software.
- *Interfaces and interrelations management*. CPS will have interfaces and interrelations everywhere, across systems, components, data, models, tools and people. System level methodologies need to deal much more explicitly with these, including their design, analysis and management.
- *Education and life-long learning*. There is also an urgent need to address these new challenges with a reshaped undergraduate education, and to implement a system for continuous professional competence training. Foundations and the T-shaping of engineers are becoming more important. Engineers increasingly need to be able to work efficiently in teams and to obtain a broader understanding than what is provided by a traditional disciplinary education (e.g. in computer science or mechanical engineering). Establishing such a broader level of understanding corresponds to the horizontal upper part of the T, see e.g. Törngren et al. (2016). Considering the speed of technology evolution, there are also strong needs to develop and adopt approaches for life-long learning, see e.g. Törngren et al. (2015), i.e. to continuously deepen and broaden the domain expert knowledge.

We emphasize that these bridging measures need to be considered in conjunction. In this sense, individual bridging measures can be seen to provide specific viewpoints for CPS development.

Relating these measures to a number of roadmaps/agendas in the area, see e.g. Platforms4CPS (2017), we find that design and architecting and MBE receive a lot of attention, while the others sometimes are not covered or covered indirectly.

This work has been motivated by the increasing complexity of CPS, and also by the desire to bridge the gap between the cyber and physical dimensions. One direction of future work is to fully expand the role of humans as part of CPS. We hope that this review of various facets of complexity of CPS will contribute to invigorate a multi-disciplinary debate on how to deal with the CPS of tomorrow!

Acknowledgements. Feedback and insights from Erik Herzog (SAAB), Martin Nilsson (RISE) and Tor Ericson (ÅF) are greatly acknowledged. We also acknowledge valuable feedback from the anonymous reviewers. This work has been partially supported by the European Commission H2020 projects Platforms4CPS and CPSE-Labs.

References

Adamsson, N.: Interdisciplinary integration in complex product development: managerial implications of embedding software in manufactured goods. Ph.D. thesis, Department of Machine Design, KTH Royal Institute of Technology, Stockholm, Sweden (2007)

Andersson, H.: Henric Andersson, senior expert, SAAB. CPS summerschool lecture, Halmstad (2017). https://www.youtube.com/playlist?list=PLRM7eLJHoNde-iM3ET-2-bHsh-KSAWLs3. Accessed Sept 2017

Axelsson, J.: On how to deal with uncertainty when architecting embedded software and systems. In: Crnkovic, I., Gruhn, V., Book, M. (eds.) ECSA 2011. LNCS, vol. 6903, pp. 199–202. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23798-0_20

Axelsson, J., et al.: Notes on agile and safety-critical development. In: XP2015 ASCS Workshop (2015)

Blackenfelt, M.: Managing complexity by product modularization. Ph.D. thesis, Department of Machine Design, KTH Royal Institute of Technology, Stockholm, Sweden (2001)

Blondel, V.D., Tsitsiklis, J.N.: A survey of computational complexity results in systems and control. Automatica **36**(2000), 1249–1274 (2000)

Boyes, H.A.: Trustworthy cyber-physical systems—a review. In: 8th IET International System Safety Conference Incorporating the Cyber Security Conference (2013)

Brooks, F.P.: No silver bullet: essence and accidents of software engineering. Computer **20**(4), 10–19 (1987)

Broy, M.: Multifunctional software systems: structured modeling and specification of functional requirements. Sci. Comput. Program. **75**(12), 1193–1214 (2010)

Broy, M., et al.: Engineering automotive software. Proc. IEEE **95**(2), 356–373 (2007)

Börjesson, F.: Product platform design – architecting methods and tools. Ph.D. thesis, Department of Machine Design, KTH Royal Institute of Technology, Stockholm (2014)

Checkland, P.: Systems Thinking, Systems Practice. Wiley, New York (2000)

Cengarle, M.V., Bensalem, S., McDermid, J., Passerone, R., Sangiovanni-Vincentelli, A., Törngren, M.: CyPhERS: Characteristics, capabilities, potential applications of Cyber-Physical Systems: a preliminary analysis, Deliverable D2.1 of the CyPhERS FP7 project, November 2013. http://www.cyphers.eu/sites/default/files/D2.1.pdf. Accessed Sept 2017

Derler, P., et al.: Modeling cyber-physical systems. Proc. IEEE Spec. Issue CPS **100**(1), 13–28 (2012)

El-khoury, J., et al.: A model-driven engineering approach to software tool interoperability based on linked data. Int. J. Adv. Softw. **9**(3 & 4), 248–259 (2016)
Engell, S., et al.: CPSoS: D3.2 Policy Proposal "European Research Agenda for Cyber-Physical Systems of Systems and their engineering needs". Report D3.2 from the EU project CPSoS (Towards a European Roadmap on Research and Innovation in Engineering and Management of Cyber-Physical Systems of Systems) (2015)
Ericson: Personnel communication with Tor Ericson, senior manager at ÅF (2017)
Eppinger, S.D., Browning, T.R.: Design Structure Matrix Methods and Applications. MIT Press, London (2012)
Eppinger, S., Salminen, V.: Patterns of Product Development Interactions. Int. Conf. on Engineering Design, ICED 01, Glasgow, August 2001
ESD: ESD symposium committee overview: engineering systems research and practice. Engineering Systems Division MIT (2003). http://esd.mit.edu/ESD_Internal_Symposium_Docs/WPS/ESD-WP-2003-01.20ESD_InternalSymposium.pdf. Accessed Sept 2017
Henzinger, T.A., Sifakis, J.: The embedded systems design challenge. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 1–15. Springer, Heidelberg (2006). https://doi.org/10.1007/11813040_1
Horváth, I., et al.: Order beyond chaos: introducing the notion of generation to characterize the continuously evolving implementations of cyber-physical systems. In: Proceedings of the ASME 2017 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Cleveland, Ohio, USA, August 2017
INCOSE: Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th edn. International Council of Systems Engineering. Wiley (2015)
Jacobson, I., Lawson, H.: Software and systems. In: Jacobson, I., Lawson, H. (eds.) Software Engineering in the Systems Context, Chap. 1. College publications (2015)
J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE Surface Vehicle Recommended Practice, September 2016
Kahneman, D.: Thinking, Fast and Slow. Penguin Books Ltd. (2012). ISBN 9780141033570
Kopetz, H.: Real-Time Systems - Design Principles for Distributed Embedded Applications. Springer, Heidelberg (2011). https://doi.org/10.1007/978-1-4419-8237-7
Kaushik, S.: Structural complexity and its implications for design of cyber-physical systems. Ph. D. thesis. MIT Engineering Systems Division, September 2014
Lawson, H.: Attaining a systems perspective. In: Jacobson, I., Lawson, H. (eds.) Software Engineering in the Systems Context. College publications (2015)
Lee, E.A.: Computing needs time. Commun. ACM **52**(5), 70–79 (2009)
Lee, E.A.: Fundamental limits of cyber-physical systems modeling. ACM Trans. Cyber-Phy. Syst. **1**, 3:1–3:26 (2016). Article no. 3
Maier, M.: Dimensions of complexity other than "complexity". In: Symposium on Complex Systems Engineering. RAND Corporation, Santa Monica, CA, 11–12 January 2007
Maier, M., Rechtin, E.: The Art of Systems Architecting. CRC Press, Boca Raton (2002)
Malvius, D.: Integrated information management in complex product development. Ph.D. thesis, Department of Machine Design, KTH Royal Institute of Technology, Sweden (2009)
McConnell, S.: Software Project Survival Guide (Developer Best Practices). Microsoft Press, Redmond (1997)
McDermid, J., Kelly, T.: Software in safety critical systems: achievement and prediction. Nucl. Future **02**(03) (2006)
Mohan, N., et al.: ATRIUM - architecting under uncertainty: for ISO 26262 compliance. In: IEEE SysCon (2017)
National Academies: A 21st Century Cyber-Physical Systems Education. National Academies of Sciences, Engineering, and Medicine. National Academies Press (2016)

NIST (2017). https://www.nist.gov/el/cyber-physical-systems. Accessed Sept 2017
Oliver, D.W., et al.: Engineering Complex Systems with Models and Objects. McGraw-Hill, New York (1996)
Platforms4CPS (2017). (see Foundations of CPS – Related Work). https://platforum.proj.kth.se/tiki-index.php?page=HomePageExternal. Accessed Sept 2017
Qamar, A.: Model and dependency management in mechatronic design. Ph.D. thesis, KTH Royal Institute of Technology, Stockholm, Sweden (2013). ISBN 978-91-7501-664-1
Qian, L., Gero, J.S.: Function-behavior structure paths and their role in analogy-based design. Artif. Intell. Eng. Des. Anal. Manuf. **10**, 289–312 (1996)
Red, E., Jensen, G., Weerakoon, P., French, D., Benzley, S.: Architectural Limitations in Multi-User Computer-Engineering Applications. Center for e-Design Publications 7 (2013). http://lib.dr.iastate.edu/edesign_pubs/7. Accessed Sept 2017
Sadigh, D., Kapoor, A.: Safe control under uncertainty with probabilistic signal temporal logic. In: Proceedings of Robotics: Science and Systems (RSS), June 2016. https://doi.org/10.15607/RSS.2016.XII.017
Sangiovanni-Vincentelli, A.: Defining platform-based design. EEDesign of EETimes (2002)
Schätz, B., et al.: Research Agenda and Recommendations for Action, Deliverable of the CyPhERS FP7 Project, March 2015
Sellgren, U.: Simulation driven design – motives, means, and opportunities. Ph.D. thesis, Department of Machine Design, KTH, Stockholm, Sweden (1999)
Shannon, C.E., Weaver, W.: The Mathematical Theory of Communication. The University of Illinois Press, Urbana (1949)
Sheard, S.: Complexity, systems and software. In: Jacobson, I., Lawson, H. (eds.) Software Engineering in the Systems Context. College Publications (2015)
Simon, H.: The Sciences of the Artificial, 3rd edn. MIT Press, Cambridge (1996)
Simmons, W., et al.: Architecture generation for moon-mars exploration using an executable meta-language, vol. AIAA-2005-6726. American Institute of Aeronautics and Astronautics (2005)
Sillitto, H.G.: On systems architects and systems architecting: some thoughts. In: Proceedings INCOSE, Singapore (2009)
Song, H., et al.: CPS: Cyber-Physical Systems: Foundations, Principles and Applications. Elsevier, New York, September 2016. ISBN 9780128038017
Steward, D.V.: The design structure system: a method for managing the design of complex systems. IEEE Trans. Eng. Manag. **EM-28** (1981). https://doi.org/10.1109/tem.1981.6448589. Accessed Sept 2017
Suh, N.P.: The Principles of Design. Oxford University Press, New York (1990)
Suh, N.P.: A theory of complexity, periodicity and the design axioms. Res. Eng. Des. **11**(2), 116–132 (1999)
Thomas Telford Journals M-W (2017). Merriam-Webster: https://www.merriam-webster.com/dictionary/cyber. Accessed Sept 2017
Törngren, M., et al.: Model based development of automotive embedded systems. In: Navet, N., Simonot-Lion, F. (eds.) Automotive Embedded Systems Handbook. Taylor and Francis CRC Press Series. Industrial Information Tech (2008)
Törngren, M., et al.: Integrating viewpoints in the development of mechatronic products. J. Mechatron. **24**(7), 745–762 (2014)
Törngren, M., et al.: Education and training challenges in the era of Cyber-Physical Systems: beyond traditional engineering. In: Workshop on Embedded and Cyber-Physical Systems Education (WESE) at ESWEEK 2015, Amsterdam (2015)
Törngren, M., et al.: Strategies and considerations in shaping cyber-physical systems education. ACM SIGBED Rev. – Spec. Issue Embed. Cyber-Phys. Syst. Educ. **14**(1), 53–60 (2016)

VDI: Design methodology for mechatronic systems - VDI 2206. VDI Guidelines, Beuth Berlin (2004)

Wagner, M., Koopman, P.: A philosophy for developing trust in self-driving cars. In: Meyer, G., Beiker, S. (eds.) Road Vehicle Automation 2. LNM, pp. 163–171. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19078-5_14

Westman, J.: Specifying safety-critical heterogeneous systems using contracts theory. Ph.D. thesis, KTH Royal Institute of Technology (2016)

Whitney, D.E.: Why mechanical design cannot be like VLSI design. Res. Eng. Des. **8**, 125–128 (1996)