Abstract—Cyber-Physical system (CPS), one of the nine key technical areas of Industry 4.0, will transform the way we interact with the physical world. After analysing the current definitions, the common challenges and hurdles faced by system designers, we show that CPS is actually a subset of a mechatronic system and suggest a new definition. We also show how system design has been approached differently by experts from different domains and outline practical difficulties in engineering CPS. As existing CPS design approaches attempt synthesis of design models at signal levels only and do not model energy interactions at physical levels, design synthesis and design automation have become difficult problems. Model-driven development holds the key to overcome design synthesis and end-to-end design automation problems.

Keywords—Cyber-Physical System (CPS); Industry 4.0; Telematics; Mechatronics System Design

I. INTRODUCTION

Industry 4.0, the fourth phase of Industrial revolution has arrived. It is fascinating, because for the first time industrial revolution has been predicted before it actually started [1]. We witnessed the first three phases powered by steam, electric and computing machines respectively. Industry 4.0 is said to describe the future phase in manufacturing, to be built and powered by a new industrial paradigm around Internet. Industry 4.0 is a collection of key terms and nine enabling technologies - Intelligent Systems, Internet Services, Social Web, Internet of Things (IoT), Big Data Solutions, Industrial Internet, and Cyber-Physical Systems (CPS) as solutions. We are ceaselessly bombarded with new buzzwords and predictions on how technology trends like Industrial IoT (IIoT), Web and Cloud Services, Wireless and Big Data have created enormous opportunities to address the future challenges to be faced by manufacturing industries. Irrespective of what buzzword or acronym one might choose, it is, in our humble opinion and as will be explained shortly, “Mechatronics wine in new bottles”.

Industry 4.0 has kick-started an innovation race with companies competing to balance performance, quality, flexibility and safety requirements. Developing “smart” products is an answer, but successful design of smart products is largely dependent on managing product complexity, heterogeneity and dynamics [2].

A. Defining Complexity

Complexity has many definitions, dimensions and metrics. It has been defined as “the measure of uncertainty in achieving the functional requirements (FRs) of a system within their specified design range” [3]. Complexity theory classifies complexity into four different types based on time dependence, periodicity, real or imaginary, and combinatorial intensity: time-independent real complexity, time-independent imaginary complexity, time-dependent combinatory complexity, and time-dependent periodic complexity [3]. A simple and practical metric of product complexity used by the aerospace and automobile community is the number of details (Part Count + Source Lines of Code) at the component level.

Defence Advanced Research Projects Agency (DARPA), the US government funded military research agency expects complexity of next-generation products to reach $1.0E + 08$ (measured as number of parts + Lines of Code), resulting in an increase of development costs by 8 to 12% per annum for aerospace products and by 4% per annum for automotive systems [6]. This also results in a proportional increase in development times as a result of increase in design, integration and testing complexity. For sustainable design, a reduction of development effort by up to 5 times will be needed for aerospace platforms. Automation of design and design processes will be the key to reducing developmental efforts.

From a system design view point, as the product functionality increases, the number of details in design descriptions increases exponentially. Fig. 1 illustrates this; if the design is viewed as a mapping process between Functional Requirements (FR) and Design Parameters (DP), the relationship between these vectors can be given by the equation

$$FR = [A] \ast DP$$  \hspace{1cm} (1)

where A is the Design Matrix that has to be synthesized and it is generally a non-quadratic matrix of constants in the
case of a linear design. In the case of a non-linear design it will be a function of DPs [7]. A small number of functional requirements $10^{6−1}$ at the “top” of the pyramid translates to $10^{7−8}$ requirements at the bottom of the design pyramid. This explosion of details at the component levels creates a “gap” between concept and implementation. The heterogeneity and interdisciplinary nature [8]–[10] of modern Mechatronic and CPS products add to the complexity. As a result, two types of complexities accrue:

(1) Product complexity as the product functionality is spread across multiple domains and energetic interactions with the environment.

(2) System complexity due to uncertainty in achieving the functional requirements (FRs) in a specified design range [3]. System Architects at the higher levels will no longer be able to see what is being done at the component levels by domain experts, as shown in Fig. 1. In addition, decomposing system functions and recomposing them to obtain “a coherent whole” becomes strenuous if not impossible. This results in lack of traceability, leading to issues like the failure to detect critical design problems at early stages of the design/development process. It also adversely affects the schedule and costs of the project.

II. THE CPS CONCEPT: OLD MECHATRONIC WINE?

CPS was recently coined in the year 2006 by Helen Gill at the US National Science Foundation [11]. Just as Internet transformed human social interaction, it is hoped that CPS will transform the way we interact with the physical worlds. In the last decade, interest in CPS has peaked world-wide due to impetus it receives from scientific and industrial communities. As a result, researchers and practitioners from many scientific disciplines have attempted to define CPS and extend domain specific theories, tools and best practices for the design and operation of CPS.

A. CPS as envisioned

As CPS has been viewed differently by researchers and practitioners, many diverse definitions for CPS exist. The control systems community defines CPS as “engineered systems whose operations are monitored, coordinated, controlled, and integrated by computing and communication components interacting with the physical environment” [12]. Meanwhile, Embedded Computing community defines CPS as “embedded intelligent ICT systems that are interconnected, interdependent, collaborative, and autonomous. They provide computing and communication, monitoring/control of physical components/processes in various applications.” [13]. Informatics community defines CPS as “... an orchestration of computers and physical systems” [14]. While traditional control theory views computing as a discrete arithmetic device, computer scientists view CPS as an extension of computing to include “the other hard to ignore effects” using “a sophisticated control-computing codesign” methodology [15]. Attempts to include biological systems have also been made. CPS is a “collection of units that bridge the cyberworld of computing and communications with the physical and biological worlds [16]”.

B. Evolution of Mechatronics

The term “Mechatronics” was reportedly coined in 1969 and has evolved rapidly (Fig. 3 and Fig. 2). From what was once considered as “feature enhancement of mechanisms using electronics”, Mechatronics as a new discipline has evolved over the years to mean “feature enhancements which cannot be otherwise obtained individually”. With the addition of Information technology in 1989, mechatronics has evolved with advances in software and Information technology [17].

Fig. 2 summarizes the multi-dimensional paradigm shifts in ICT influencing Mechatronic and CPS product development. Evolution is shown along the x-axis; the arrow (orange color) represents the increase in complexity as a result of evolution of the design process and the evolving nature of the technology domains and domain descriptions. Non-functional design requirements, such as reusability, executability of models and modularity keep increasing along the y-axes. From a software perspective, a paradigm shift from “Programming” to “Models” has happened, which has, to a great extent, helped alleviate the inability of 3rd Generation Languages (3GL) to represent design functionality effectively [18]. Computer languages have also evolved rapidly and we now have languages and scripts that can be used to create entire applications and frameworks. Similarly, platforms have evolved, and a variety of platforms...
(.NET, Java etc.) to implement the developed applications are available. This change requires that the languages and models be unified and generic, leading to a shift from “code” and “procedures” to “modeling”. Software modeling frameworks have also evolved from Object-Oriented, Component-Oriented, Model Driven, Services-Oriented and Actor-Oriented Frameworks to Knowledge Oriented Frameworks and have been promptly exploited by Mechatronic system designers [19].

Tight coupling of physical systems to embedded controllers in the year 2000’s gave way to loosely coupled, net-worked physical systems. By the year 2020, it might even graduate to become “Internet of everything”.

C. CPS as a Mechatronic subsystem

Hence, our choice is to use the term Mechatronics in a broader context and CPS to mean a subset of mechatronics systems with dynamic coupling between physical and computing domains. We argue that CPS by definition (Fig. 3) and effect, mean Mechatronic products and systems. We are not alone. For instance Neema et.al [20] state that “CPS are mechatronic systems, characterized by tight integration between computational, communicational and physical components”. In the context of this paper, as will be seen later, the term CPS has been used specifically to refer to mechatronics systems that have dynamically coupled interaction with the environment at a physical level, both through energy and signal interactions.

In summary, CPS is a new breed of complex, “software-intensive” mechatronic systems characterized by tight integration and real-time interaction between computational, communicational and physical components and sub-systems. CPS calls for a close coupling between the physical and computing domains. The focus is on getting the physics right; the rest is mathematics and computation [19].

III. THE DESIGNERS PREDICAMENT

There is a relentless push to create faster, smarter and better products and systems. Designers bear the brunt of this shift. They face the mammoth task of not only closing the gap between “what is imagined” and “what is created” but also shouldering the responsibility of balancing conflicting requirements such as: higher quality but low cost; more features but less time-to-market; optimizing resources without compromising on collaboration among multidisciplinary teams, etc.

The major problem Mechatronics and CPS designers face currently is that there are no general purpose frameworks for architectural exploration, design synthesis and design automation [17], [23]. Only single-domain frameworks for design automation are available: CORBA middleware / MOF for software & information technology; Simulink/ Stateflow for control systems; System-C for embedded systems etc. But successful CPS design requires the integration of modules from multiple domains to bridge the cyber world of computing & communications and the physical worlds [16]. Today, CPS integration is more of a management art than science [24], and without a rigorous unifying framework, integration of systems and models remains ad-hoc [25]. In addition, most CPS design approaches attempt synthesis of design models at signal levels only and do not model energy interactions at physical levels. This renders design synthesis and design automation difficult.

Mechatronics (and CPS) naturally inherit complexity due to concurrent, dynamic, real-time interactions among heterogeneous components from both cyber and physical domains (mechanical, electrical, communication, real-time control etc.). An important design tenet of mechatronic systems (and CPS) in particular, is that they should be “designed as a whole”. Therefore constant evaluation and adjustment of the relationships between the parts and the whole is central to the design process. This is called Architectural Exploration and Evaluation, and requires a conscious radical shift from pure “divide and conquer” analytical approaches. Hence Synthesis is more suitable for the design of Mechatronics and CPS, and should be used predominantly to combine disparate components and subsystems. However, synthesis (opposite of analysis) is non-trivial. Analysis is easier, as it only involves breaking down system functionality into modules for analysis. Synthesis on the other hand, involves not only decomposition of system functions but also recomposition and integration of the models to obtain a coherent whole. However synthesis as imagined is different and far from reality [26].

IV. THE HURDLES

There are many practical challenges in design synthesis and design automation. Here, they can be viewed from two perspectives: (a) General challenges common to all synthesis and automation endeavors. (b) Technical challenges specific to design synthesis and design automation of mechatronic systems and CPS.

A. General Challenges - Synthesis and Automation

From a general perspective, major challenges common to all synthesis endeavors are: composition of complex models from simple models, composability and availability of domain independent frameworks for design synthesis and design automation. Selecting the appropriate models for synthesis is a challenging task for the designer. A typical complex system has many models (structural, behavioral, functional etc.) expressed in various viewpoints. They need to be transformed and synergistically combined to obtain a “coherent whole”. The difficulty is compounded by the fact that models are abstract by definition, hide information, and are not absolute representations of reality. Multiple models for the same system can exist based on purpose and viewpoints of modelers. Models

1Software-intensive systems are those complex systems where software contributes essential influences to the design, construction, deployment and evolution of the system as a whole.

2Architectural is defined in IEEE 1471 standard as “the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution” [21].

3Synthesis is one of the three design methods where parts, elements or individual findings are combined/fitted/associated to create a more useful working combination of whole new working system [22, pg.59].

4Design Automation is the application of computational tools to perform tasks (directly or indirectly related to design of components, systems or processes), that were once performed by human beings or tasks that are impossible to be performed by human beings due to the huge number of design details.

5Analysis is the most popular of the three design methods and involves breaking down complex elements into smaller elements to study of these elements in isolation and observe their interrelationships [22, pg.58]

6Composability is the ability to link subsystems such that subsystem properties are held at system levels [26].
for simulation are structurally and functionally different from models for design synthesis. Design requirement analysis also results in functional models which are different from behavior or structural models. Frequently, a single function is spread across multiple domains and subsystems. Before integration, models have to be abstracted, unified and transformed for proper composition or else a "whole" which is more than the sum of the parts will result. Hence a unifying framework is needed to integrate various models and conserve model properties selected for synthesis.

Frameworks\(^8\) or problem-solving approaches have traditionally been used for development of software-intensive systems. Concept-to-Code (C2C) frameworks which provide complete design automation for engineering systems are not available. Hence frameworks have to be developed for architectural exploration and seamless integration of CPS models as a system of interacting subsystems.

Existing Model-Driven Development (MDD) frameworks like Model-Driven Architecture (MDA) do not support design synthesis and automation completely. Model Transformation\(^9\) is the heart and soul of MDD [28]. MDA (Object Management Group’s version of MDD) prescribes transformation through 3 stages: Computationally Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). Successful model transformations across all three levels result in complete design automation, also known as C2C frameworks. Currently, model transformation (from PIM to PSM and PSM to Code using automatic code generation tools) has been successfully attempted by industry and researchers using popular tool chains. But the gap from CIM to PIM has not been bridged [29], preventing design automation and C2C implementations. Frameworks for CPS should support design automation and architectural exploration.

B. Technical challenges - Design Automation of Mechatronic and CPS

From a technical perspective, lack of "scientific foundations", "theoretical glue" and "system integration approaches" are the major obstacles for design automation and synthesis. Requirements such as real-time communication and concurrency, and passing of constraints and dynamics across subsystems pose problems for synthesis.

Currently, physical processes are controlled and monitored using discrete (embedded and network) controllers which use feedback loops to effect computations, whereas in reality the physical world is continuous in time and also concurrent. Existing computing models and abstractions do not model these inexorable properties [30]. For example, even if we have deterministic subsystem models (e.g., differential equations, synchronous digital logic, etc.) there is no guarantee that the composed system model will preserve determinism. Absence of model-driven methods for precise and predictable orchestration is a major challenge [24]. These foundational challenges call for a top-to-bottom rethinking of computation [30] and Models of Computation (MoC).

System Integration of CPS is a "grand challenge" due to lack of scientific theoretical foundation and the difficulty in understanding the science of System Integration [24]. Managing multi-disciplinary system integration is widely recognized as a very difficult task [31–35] and is not supported by the existing frameworks and tools [32]. From a systems integration perspective, absence of science and technology foundations for CPS integration and inadequate computing foundations [30] required to balance loosely coupled subsystems and their tightly coupled dynamic interactions makes system integration a non-trivial problem. Researchers have called for a new science and technology foundation, and rapid progress is to be made for CPS integration based on precise and predictable models [24]. However, in multi-disciplinary product design environments such as CPS, cross-disciplinary problems that are hard to predict and solve usually occur in the last stages of the engineering design process. Frequently, subsystem interactions which are very difficult to assess or predict early are major causes of design failures [31]. From an information technology perspective, component-based software technologies such as object-oriented design have helped in the development of large systems. However, they are deeply sequential and the imperative computational models cannot adapt to concurrent real-time computing [30]. Actor-oriented [36] and model-based design techniques [26] have been suggested as better alternatives and are still under development.

V. Conclusion

Industry 4.0 has kick-started a growing product development trend, which tends to blend electro-mechanics, digital electronics, distributed control, intelligence, the internet, and cloud computing elements into a single integrated system and popularly called Mechatronics system until recently. A sub-set of mechatronic products and systems — when characterized by concurrent, dynamic and real-time interactions among heterogeneous subsystems from “Cyber” and “Physical” domains — are now being increasingly called Cyber-Physical systems (CPS). The synthesis of these increasingly powerful and complex products is a difficult proposition, as it involves integrating a large number of elements from many engineering disciplines, with strongly coupled physical energy fields, having little or no cross-functional understanding of emergent system behavior. CPS, like Mechatronics systems, need to be designed as a whole and currently, no general purpose framework is available for architectural exploration, design synthesis and system integration.

Model-Driven Development (MDD), a popular software engineering approach, has been successfully used by designers from the embedded software and informatics communities. Even though MDD is no panacea, we argue that MDD is the most suitable model-centric design approach for achieving end-to-end design automation. Some initial MDD framework implementations [6], [19] for CPS design synthesis and design automation have been successful. However, three major impediments have to be overcome. Firstly, we need to ensure that model representation is formal enough and embellished with sufficient information, thereby guaranteeing automatic transformation into other detailed models and finally into codes. Secondly, as CPS and Mechatronics system models are highly dynamic and behavior-intensive, they should be modeled as

---

\(^8\) Abstraction is hiding/removing irrelevant details in a view point [27].

\(^9\) Framework is a popular software approach which provides a way to overcome limitations in technology.

\(^9\) Model Transformation: Process of converting a model to another form suitable for execution, without loss of semantics.
energetic, continuous-time models at the highest (Computationally independent (CIM)) levels. For functional integration at lower levels, they can be, if required, transformed to signal based discrete-time engineering system models. Thirdly, when keeping the framework flexible and open, it becomes vendor-neutral and can accommodate nascent and evolving design process methodologies such as VDI2206.

ACKNOWLEDGMENT

This work was supported by Monash University and MOHE grant FRGS/1/2015/TK08/MUSM/02/1, titled: Towards a model synthesis framework for conceptual modeling of cyber-physical systems.

REFERENCES


