Challenges, Promising Solutions and Open Problems of Cyber-Physical Systems

Lichen Zhang*, Jifeng He and Wensheng Yu
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University, Shanghai 200062, China
zhanglichen1962@163.com

Abstract

Cyber physical systems(CPS) include a lot of high complexity computing such as physical aspect modeling, dynamic analysis and verification of continuous dynamic property, analysis and verification of real-time property, analysis and verification of spatial property, scheduling and non-functional requirements. The correctness of computing results of cyber physical systems not only meets the time requirements, but also meets the spatial requirement, this make that it is impossible to solve the several difficult computing problems by traditional methods. In this paper, we describe some of the research directions that we are taking toward addressing some of the challenges involved in building cyber physical systems. The ultimate goal is to develop foundations and techniques for building safe and effective cyber physical systems.

Keywords*: Cyber Physical Systems, Continuous Dynamic, Temporal-Spatial Requirements, Non-functional requirements*

1. Introduction

Cyber physical systems [1] are dynamical systems with interacting continuous-time dynamics and discrete-event dynamics. Hybrid behavior arises in many contexts, both in man-made systems and in nature. Continuous systems such as walking robots, flight systems, train systems and process control systems, are well-suited to be modeled as cyber physical systems. In these examples, the continuous dynamics model system motion or chemical reactions, while the discrete dynamics model the sequence of contacts or collisions in the gait cycle, cell divisions, valves and pumps switching, control modes, and coordination protocols. The system dynamics are complex enough that traditional analysis and control methods based solely on differential equations are not computationally feasible. To understand the behavior of cyber physical systems, to specify, to analyze, to design, to verify, and to control these systems, theoretical advances and analytical tools are needed.

Cyber physical systems, due to their increased size and complexity relative to traditional embedded systems, present numerous developmental challenges. The long-term viability of requires addressing these challenges through the development of new design, composition, verification, and validation techniques. These present new opportunities for researchers in cyber physical systems. In this paper, we describe some of the research directions that we are taking toward addressing some of the challenges involved in building cyber physical systems. The ultimate goal is to develop foundations and techniques for building safe and effective cyber physical systems.

* Corresponding Author

2. Research Challenges of Cyber Physical Systems

The complexity of future cyber-physical systems is beyond the state of current theory and practice in reasoning about time and space; dynamic analysis and verification of continuous property; temporal-spatial scheduling; non-functional requirements such as security [2], fault tolerance and safety [3].

2.1 Physical aspect modeling in Cyber Physical Systems

The design of complex cyber physical systems, such as air and spacecraft, automobiles, and power plants relies increasingly on the use of system models that describe the physical system and its controller. The physical system model is inherently an abstract representation of the physical system dynamics, where the level of abstraction is guided by how the model is to be used. To manage the complexity of creating models of large systems, representation schemes are needed to help the modeler focus on the behavior at the right level of abstraction. A good modeling tool must use a compositional approach that allows the modeler to mimic the task of physically constructing a system from its parts. Typically physical cyber physical show some of the following features:

- mixed-domain (mechanical, electrical, thermal, fluidic, *: : :* phenomena),

- partially close coupling between these domains, side effects, cross coupling,

- distributed and lumped (concentrated) elements,

- discrete and continuous signals and systems (in electronics: analog and digital),

- very large and stiff systems of differential equations to describe the continuous subsystems.

Depending on the level of abstraction, partial differential equations (PDE) and ordinary differential equations (ODE) are the mathematical models (system equations) of continuous subsystems. Therefore, the following features are required of a general modeling language [4]:

• Since the system of interest may contain physical parts, the language must support modeling of continuous behavior.

• It should be possible to describe interfaces between physical components that are not restricted to spatial adjacency or flows of information, but also allow flows of energy and material.

• Some parts may be software, which is strictly logical, thus the language must be able to support discrete behavior as well, including behavior where the structure changes dynamically, as may be the case for software.

• It should allow efficient modeling, in the sense of short time to construct models, in order to reduce lead times, and must therefore support reuse of models.

Cyber physical systems pose the following Challenges in physical aspect modeling:

• Variables whose values change continuously as functions of time are needed to represent the current state of a physical system.

• Equations that relate the variables to each other, describing the dynamic behavior of the system.

• Physical laws are mostly differential equations, so a way of referring to the current time and to the derivative of expressions is needed.

• Realistic physical systems have different modes, i.e., they behave differently, depending on whether certain variables have values in specific ranges. Ways of describing the modes, the relations that hold in each of them, and the transitions between them, are needed. It should also be possible to describe failure modes, where components show abnormal behavior.

• It should be possible to describe the interfaces of physical components, including continuous flows of energy and matter.

• To construct large models, convenient ways to compose models of parts from libraries are needed. Preferably, common relations that are required to specify the nature of connections should not have to be restated every time two object interfaces are connected.

2.2 Space and Time

Cyber physical systems are spatio-temporal [5] in the sense that correct behavior will be defined in terms of both space and time. As cyber physical systems, computations must be completed within established response times, for which we will need new notions of timeliness [6], but they may also have varying temporal requirements based on the current frame of time. As spatial systems [7], the computations performed and their timeliness will be dependent on

(i) the location of the platform in its environment,

(ii) the velocity with which the platform is moving,

(iii) the number of objects in the environment, and

(iv) the velocity vectors of the objects in the environment.

The existing theory for describing objects in both space and time is not applicable to mobile cyber-physical systems for at least three reasons: (1)the theory ignores physical attributes and constraints on the objects; (2) the interaction of objects in space in time is not addressed; and (3) the range and precision of time resolution is not adequate for real-time**.**

Time is fundamentally different from the state components of a discrete computing system. For all we know, time is continuous, monotonic, and divergent, and program variables generally happen not to have either one of these characteristics [8]. Timing analysis attempts to determine bounds on the execution times of a task when executed on a particular hardware. The time for a particular execution depends on the path through the task taken by control and the time spent in the statements or instructions on this path on this hardware. Accordingly, the determination of execution-time bounds has to consider the potential control-flow paths and the execution times for this set of paths. A modular approach to the timing-analysis problem splits the overall task into a sequence of subtasks. Some of them deal with properties of the control flow, others with the execution time of instructions or sequences of instructions on the given hardware. There are two different classes of methods.

Static methods. These methods do not rely on executing code on real hardware or on a simulator. They rather take the task code itself, maybe together with some annotations,

analyze the set of possible control flow paths through the task, combine control flow with some (abstract) model of the hardware architecture, and obtain upper bounds for this combination.

Measurement-based methods. These methods execute the task or task parts on the given hardware or a simulator for some set of inputs. They then take the measured times and derive the maximal and minimal observed execution times, or their distribution or combine the measured times of code snippets to results for the whole task. Static methods emphasize safety by producing bounds on the execution time, guaranteeing that the execution time will not exceed these bounds. The bounds allow safe schedulability analysis of hard real-time systems.

Worst case execution time analysis computes upper bounds on the execution time of tasks in a system. As the general problem of determining the WCET of an arbitrary piece of code is undecidable (it is very similar to the halting problem) one can not expect to obtain the exact value, instead, only an upper bound can be provided. However, it is desirable for the WCET to be as tight as possible. WCET analysis is usually done at two levels. The low-level, which is done at the object code, considers the effects of hardware level features (like cache and pipelining) On the other hand, high level analysis is performed at the source code and it focuses on characterizing possible execution paths. It usually relies on annotations provided by the user to describe execution frequencies (like maximum number of loop. One need contribute by providing support for predictable AOSD, by enabling WCET analysis of aspects, components, and the resulting aspect-oriented software (when aspects are weaved into components). The WCET analysis for AOSD is based on the symbolic WCET analysis. We need a new algorithm for calculating high-level WCETs of aspect-oriented software.

The computer industry is rapidly moving towards the multi-core processors. Compared to uniprocessors, multi-core chips offer a significant boost in processing capability while consuming less energy and less board space. Therefore, multi-core processors can potentially benefit future high-performance cyber-physical systems. For example, from the viewpoint of time predictability, segregating different real-time tasks onto different CPUs makes it easier to determine whether they'll meet their deadlines. Also, fault tolerance is typically vital for safety-critical CPS such as fly- and drive-by-wire systems, and a multicore processor can cost-effectively improve reliability by exploiting redundant cores that are tightly coupled on the same chip. Multi-core processors, however, are generally designed for improving the average-case throughput, which will further complicate the WCET analysis as compared to uniprocessors, mainly due to the possible interferences from other co-running threads in using the shared resources, for instance the shared caches and memory, which can significantly increase the worst-case execution time (WCET).

2.3 Complex Continuous Dynamics

The dynamic behavior of cyber physical systems is captured in hybrid models [9]. In a manufacturing process, for example, parts may be processed in a particular machine, but only the arrival of a part triggers the process; that is, the manufacturing process is composed of the event-driven dynamics of the parts moving among different machines and the time-driven dynamics of the processes within particular machines. Frequently in hybrid systems in the past, the event-driven dynamics were studied separately from the time-driven dynamics, the former via automata or Petri net models and the latter via differential or difference equations. To understand fully the system’s behavior and meet high performance specifications, one needs to model all dynamics together with their interactions. For example, Flight calculations

are made quite precisely for space missions, taking into account such factors as the Earth's oblateness and non-uniform mass distribution; gravitational forces of all nearby bodies, including the Moon, Sun, and other planets; and three-dimensional flight path.

Reachability questions for systems with complex continuous dynamics are among the most challenging problems in verifying cyber physical systems. Cyber physical systems are models for these systems with interacting discrete and continuous transitions, with the latter being governed by differential equations. For simple systems whose differential equations have solutions that are polynomials in the state variables, quantifier elimination can be used for verification. Unfortunately, this symbolic approach does not scale to systems with complicated differential equations whose solutions do not support quantifier elimination (*e.g.*, when they are transcendental functions) or cannot be given in closed form. Numerical or approximation approaches can deal with more general dynamics. However, numerical or approximation errors need to be handled carefully as they easily cause unsoundness. Thus, numerical approaches can be used for falsification but not (ultimately) for verification.

2.4 Non-Functional Requirements

Non-functional requirements[10] address important issues of quality and restrictions for cyber physical systems, although some of their particular characteristics make their specification and analysis difficult: firstly, non-functional requirements can be subjective, since they can be interpreted and evaluated differently by different people; secondly, Non-functional requirements can be relative, since their importance and description may vary depending on the particular domain being considered; thirdly, non-functional requirements can be interacting, since the satisfaction of a particular non-functional requirement can hurt or help the achievement of other non-functional requirement. Dependability is that property of a system that justifies placing one's reliance on it. The dependability of a system is the collective term that describes the availability performance of a system and its influencing factors: reliability, safety, maintainability and maintenance support performance. These non-functional properties are highly important for real-time systems as they are designed to operate in environments where failure to provide functionality or service can have enormous cost both from financial, influential or physical aspects. Therefore it is essential that these properties are calculated as precisely as possible during the design and operation of such systems. Reliability is the ability of a system or component to provide its required functionality or services under given conditions for a specified period of time. Availability is the ratio of total time that a system or a component is functional during a specified period and the length of the period. Maintainability can be specified as the probability that a component or system will be restored to a given condition within a period of time. Safety is described as the absence of serious consequences on the user or environment in case of failure. Safety can be defined as "a property of a system that it will not endanger human life or the environment" .A system is safety-critical if safety cannot be ensured when it fails to provide correct service. Integrity can be specified as the absence of improper alterations on the target system or component. Survivability can be defined as the ability of the system to remain functional after a natural or man-made disturbance.

3. Solutions to Cyber Physical Systems Research Challenges

A number of the methodology and tools have been proposed for the design of cyber physical systems. However, they are still evolving because systems and software engineers do not yet completely understand the cyber physical system requirements, and the complex

dynamic behavior of cyber physical systems is difficult of capture in a simple understandable representation. In our opinion, an acceptable cyber physical system design methodology must synthesize the different views of systems, use a number of different methods, and consist of an orderly series of steps to assure that all aspects of the requirements and design have been considered. We propose a design methodology of cyber physical systems that is different from current methodologies and serves to help manage the complexity of cyber physical systems. One of the distinguishing aspects of this methodology is in its ability to express timing constraints and spatial constraints and verify if such constraints are met. A second distinguishing aspect of this methodology is that it can decompose the life cycle into several stages based on the Model based development according to different views of systems, and we use different methods to modeling different aspects of cyber physical systems.

Physical aspect modeling in cyber physical systems: Modelica [11] is a new language for hierarchical object oriented physical modeling which is developed through an international effort. The language unifies and generalizes previous object-oriented modeling languages. The language has been designed to allow tools to generate efficient simulation code automatically with the main objective to facilitate exchange of models, model libraries and simulation specifications. Modelica is primarily a modeling language, sometimes called hardware description language that allows the user to specify mathematical models of complex physical systems, e.g. for the purpose of computer simulation of dynamic systems where behavior evolves as a function of time. Modelica is also an object-oriented equation based programming language, oriented towards computational applications with high complexity requiring high performance.

SysML [12] is a standardized general purpose graphical modeling language for capturing complex system descriptions in terms of their structure, behavior, properties, and requirements.

Integrating the descriptive power of SysML models with the analytic and computational power of Modelica models provides a capability that is significantlygreater than provided by SysML or Modelica individually. SysML and Modelica are two complementary languages supported by two active communities. By integrating SysML and Modelica, we combine the very expressive, formal language for differential algebraic equations and discrete events of Modelica with the very expressive SysML constructs for requirements, structural decomposition, logical behavior and corresponding cross-cutting constructs. In addition, the two communities are expected to benefit from the exchange of multi-domain model libraries and the potential for improved and expanded commercial and open-source tool support. The profile of integrating Modelica and SysML supports modeling with all Modelica constructs and properties, *i.e.*, restricted classes,equations, generics, variables, etc. Using the diagrams of the Modelica and SysML, it is possible to describe most of the aspects of a system being designed and thus support system development process phases such as requirements analysis, design, implementation, verification, validation and integration. The profile of integrating Modelica and SysML supports mathematical modeling with equations since equations specify behavior of a system. Simulation diagrams are introduced to model and document simulation parameters and simulation results in a consistent and usable way.

The timing analysis: In timing analysis phase, the specification of cyber physical systems is first extended with time annotation to express with the timing requirements, that is the specification is annotated with task parameters: worst-case execution time, deadline and period, etc. Then specification is mapped to the precedence graph according to mapping rules. Task parameters together with the precedence graph are used to perform timing analysis, that

is, allocation and schedulability analysis according to scheduling algorithms. We believe that you must demonstrate that the requirements are met at two levels of detail and for each class of task. At the macroscopic (system-wide) level we need to show first that all critical tasks will always make their deadline and that all noncritical tasks meet overall requirements. Alternatively, the requirements might be to maximize the value of noncritical hard and soft real-time tasks executed by the system. At the microscopic level, we also would like some level of predictability. For the critical tasks we already know that they will always make their deadline. For the other real-time tasks their performance will depend on the current state of the system.

The specification of a UML™ profile adds capabilities to UML for model-driven development of Real Time and Embedded Systems (RTES). This extension, called the UML profile for MARTE [13] (in short MARTE for Modeling and Analysis of Real-Time and Embedded systems), provides support for specification, design, and verification/validation stages. This new profile is intended to replace the existing UML Profile for Schedulability, Performance and Time. The time domain model described in MARTE identifies the set of time-related concepts and semantics that are supported by this profile. The model is quite general, and a given application may need to use only a subset of its proposed concepts and semantics. Time can be differently perceived at the different phases of the development of an embedded real-time system (modeling, design, performance analysis, schedulability analysis, implementation, *etc*.). The concept of ordering (*i.e.*, something occurring before or after another thing) is common to many Time representations. The Clock Constraint Specification Language (CCSL) [14], initially specified in an annex of MARTE, offers a general set of notations to specify causal, chronological and timed properties on these models and has been used in various subdomains. CCSL is formally defined and CCSL specifications can be executed at the model level. CCSL is based on the notion of *clocks* which is a general name to denote a totally ordered sequence of event occurrences, called the *instants* of the clock. CCSL is a high-level multi-clock language and the original semantics does not require totally ordered models.

Spatio-temporal reasoning: The main aim of spatio-temporal research is to introduce a hierarchy of languages intended for qualitative spatio-temporal representation and reasoning, provide these languages with topological temporal semantics construct effective reasoning algorithms and estimate their computational complexity. There exist a number of qualitative constraint calculi that are used to represent and reason about temporal or spatial configurations. However, there are only very few approaches aiming to create a spatio-temporal constraint calculus. We use the spatial calculus RCC-8 and Allen's interval calculus in order to construct a qualitative spatio-temporal calculus. When adding the restriction that the size of the spatial regions persists over time, or that changes are continuous, the calculus becomes more useful, but the satisfiability problem appears to be much harder.

Temporal-Spatial Scheduling: Spatial requirements need to be guaranteed in cyber physical systems such as transport control systems and flight control systems in addition to timing constraints. Unfortunately, most conventional scheduling algorithms only take one dimension of them into account [15]. We propose a scheduler model that can be used for multi-dimensional scheduling according spatio-temporal constraints. Based on the scheduler model, we propose a heuristic multi-dimensional scheduling strategy, consisting of two steps. The first step is to select one scheduling algorithm according to spatial requirements. In step 2, we use the selected select one scheduling algorithm to scheduling tasks in cyber physical systems according to timing constraints.

Continuous Dynamics: The standard approach to dealing with continuous dynamics for cyber physical systems is to use symbolic or numerical solutions of their respective differential equations. Unfortunately, the range of systems that is amenable to these techniques is fairly limited, because even solutions of simple linear differential equations quickly fall into undecidable classes of arithmetic. As a means for verifying cyber physical systems with challenging continuous dynamics without having to solve their differential equations, A solution is to complement discrete induction for loops and repetitions with a new form of differential induction for differential equations. Differential induction is a natural induction technique for differential equations. It is based on the local dynamics of the (right-hand side of the) differential equations themselves and does not need closed-form solutions for the differential equations. Because differential equations are simpler than their solutions (which is part of the representational power of differential equations), differential induction techniques working with the differential equations themselves are more scalable than techniques that need solutions of differential equations. The differential induction techniques even generalize to differential-algebraic constraints with differential inequalities or quantifiers in the dynamics. To further increase the verification power, we add differential strengthening or differential cuts as a powerful proof technique for refining the system dynamics with auxiliary invariants that can simplify the proof of the original property significantly. The basic insight is that auxiliary properties that are provable invariants of the dynamics can help prove the original property even if it was not provable before.

Non-Functional Requirements: Aspect-oriented software development methods make up object-oriented software development methods in system development needs of non-functional characteristics of the existing limitations question problem [16]. Use separate technology of concerns separates all the crosscutting concerns of the system, and then analyzed, designed, modeled for each cross-cutting concerns, to address crosscutting concerns in object-oriented software development, the code tangling and scattering problems, enhancing the system's modular degree, lowering coupling between modules.

4. Conclusion

In this paper, we described some of the research directions that we are taking toward addressing some of the challenges involved in building cyber physical systems. The ultimate goal is to develop foundations and techniques for building safe and effective cyber physical systems.

The further work is devoted to develop the analysis, design and verification methods for cyber physical systems.

Acknowledgments

This work is supported by Shanghai 085 Project for Municipal Universities and the Innovation Program of Shanghai Municipal Education Commission under grant No. ZF1213, national high technology research and development program of China (No.2011AA010101), national basic research program of China (No.2011CB302904), the national science foundation of China under grant No.61173046, No.61021004, No.61061130541, No.91118008), doctoral program foundation of institutions of higher education of China (No. 20120076130003), national; science foundation of Guangdong province under grant (No.S2011010004905).

References

[1] E. A. Lee and S. A. Seshia, "Introduction to Embedded Systems – A Cyber-Physical Systems Approach", Berkeley, CA, LeeSeshia.org, **(2011)**.
[2] F. Y. Sattarova and T. -h. Kim, "IT Security Review: Privacy, Protection, Access Control, Assurance and System Security", International Journal of Multimedia and Ubiquitous Engineering, vol. 2, no. 2, 17-32, **(2007)** April.
[3] M. B. Swarup and P. S. Ramaiah, "A Software Safety Model for Safety Critical Applications", International Journal of Software Engineering and Its Applications, vol. 3, no. 4, **(2009)** October, pp. 21-32.
[4] J. Axelsson, "Model based systems engineering using a continuous-time extension of the Unified Modeling Language (UML)", onlinelibrary.wiley.com/doi/10.1002/sys.10021/pdf.
[5] A. Gerevini and B. Nebel, "Qualitative Spatio-Temporal Reasoning with RCC-8 and Allen's Interval Calculus: Computational Complexity", Technical Report, DEA – University of Brescia, **(2002)** May.
[6] K. -D. Kwon, M. Sugayam and T. Nakajima, "KTAS: Analysis of Timer Latency for Embedded Linux Kernel", International Journal of Advanced Science and Technology, vol. 19, **(2010)** June, pp. 59-70,
[7] D. A. Randell, Z. Cui and A. G. Cohn, "A spatial logic based on regions and connection", Proc. KR-92, **(1992)**, Morgan Kaufmann, pp. 165–176.
[8] R. Alur and T. A. Henzinger, "Real-time logics: Complexity and expressiveness", in Proc. of the 5th Annual IEEE Symp. on Logic in Computer Science, **(1990)**.
[9] A. Platzer, "Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics", Springer, **(2010)**, pp. 426, ISBN 978-3-642-14508.
[10] Wehrmeister, *et al.*, "An Aspect-Oriented Approach for Dealing with Non-Functional Requirements in a Model-Driven Development of Distributed Embedded Real-Time Systems", In: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, **(2007)** May 7-9, pp. 428-432, IEEE Computer Society, Los Alamitos.
[11] P. Fritzson, "Principles of Object-Oriented Modeling and Simulation with Modelica 2.1", New York, NY: Wiley-IEEE Press, **(2004)**.
[12] T. A. Johnson, C. J. J. Paredis and R. M. Burkhart, "Integrating models and simulations of continuous dynamics into SysML", 6th International Modelica Conference, Modelica Association, Bielefeld, Germany, **(2008)**, pp. 135-145.
[13] OMG, "UML Profile for MARTE, v1.0", Object Management Group, **(2009)**, formal/2009-11-02.
[14] C. Andr′e, "Syntax and semantics of the clock constraint specification language", Technical Report 6925, INRIA, **(2009)**.
[15] D. Virmani and S. Jain, "Real Time Scheduling for Wireless Sensor Networks", International Journal of Hybrid Information Technology, vol. 5, no. 1, **(2012)** January, pp. 61-68.
[16] X. Wen and H. Yu, "Real-Time Systems Modeling and Verification with Aspect-Oriented Timed Statecharts", International Journal of Hybrid Information Technology, vol. 5, no. 1, **(2012)** January, pp. 193-198.