Industrial Cyber-Physical Systems — iCyPhy

Amit Fisher, Clas A. Jacobson, Edward A. Lee, Richard M. Murray, Alberto Sangiovanni-Vincentelli, and Eelco Scholte

Abstract ICyPhy is a pre-competitive industry-academic partnership focused on architectures, abstractions, technologies, methodologies, and supporting tools for the design, modeling, and analysis of large-scale complex systems. The purpose of this partnership is to promote research that applies broadly across industries, providing the intellectual foundation for next generation systems engineering. The focus is on cyber-physical systems, which combine a cyber side (computing and networking) with a physical side (e.g., mechanical, electrical, and chemical processes). Such systems present the biggest challenges and biggest opportunities in several critical industrial segments such as electronics, energy, automotive, defense and aerospace, telecommunications, instrumentation, and industrial automation. The approach leverages considerable experience designing complex artifacts in the semiconductor, embedded systems, and software industries, and major recent advances in algorithmic techniques for dealing with complexity. This consortium adapts and extends these techniques to handle the fundamentally different challenges in large-scale cyber-physical systems.

Amit Fisher
IBM, Sunnyvale, CA, USA, e-mail: amitf@il.ibm.com

Clas Jacobson
UTRC, East Hartford, CT, USA e-mail: jacobsCA@utrc.utc.com

Edward A. Lee
UC Berkeley, Berkeley, CA, USA e-mail: eal@eecs.berkeley.edu

Richard Murray
California Institute of Technology, Pasadena, CA, USA e-mail: murray@cds.caltech.edu

Alberto Sangiovanni-Vincentelli
UC Berkeley, Berkeley, CA, USA e-mail: alberto@eecs.berkeley.edu

Eelco Scholte
UTC Aerospace Systems, Windsor Locks, CT, USA e-mail: Eelco.Scholte@utas.utc.com

In Proc. Complex Systems Design & Management (CSD&M), Paris, France, Dec. 4-6, 2013, Springer.

1 Industrial Motivation

Efficient and effective design of distributed multi-scale complex systems is largely an unsolved problem. Complex systems are compositions of heterogeneous components, which for cyber-physical systems often include electromechanical, chemical, thermal, computing, and communication elements. These subsystems are interconnected, often uncertain in specification, and encompassing environmental effects. The dynamics of all the elements — both cyber and physical — are critical to the performance of the overall system. Such systems are software and network enabled, and there is significant cost and schedule pressure during development. The technology drivers causing the change in delivery are the pervasive use of electronic control units, and consequently of communication networks, and the blurring of distinctions between software, firmware, hardware and multi-physics systems. These drivers are creating the possibility for placing vastly more functionality into products, but at the same time increase interconnectivity at the risk of unwanted system interactions found late in the development process.

To solve this problem we need a rigorous approach to systems engineering, specifically a methodology for product system level design, optimization and verification that:

- Provides guarantees of performance and reliability against customer requirements while achieving cost and time-to-market objectives;
- Produces modular, extensible architectures for products incorporating electromechanical components, embedded electronic systems, wired and wireless communication networks and application software;
- Exploits analytical tools and techniques to determine design choices and ensure robust system performance despite variations caused by product manufacturing, integration with other products and customer operation; and
- Achieves these objectives through the coordinated execution of a prescriptive, repeatable and measurable process.

Yet industry is still far from developing and using such a systems engineering methodology. Indeed there are no rigorous foundations in systems engineering that can address the issues of the overall design flow, and no analysis and synthesis tools for the design and verification of highly distributed systems. Consequently systems engineering practice is often a collection of common-sense, heuristic approaches based on experience and use of legacy designs. There have been advances in the domain of systems engineering science in academia, in some industrial segments such as automotive, and in some tool companies, but the overall knowledge of these advances and of their potential in the system industry is at best spotty.

This paper gives the motivation and goals for an industry-academic partnership called Industrial Cyber-Physical Systems (iCyPhy) that is addressing these challenges. ICyPhy was formed in December of 2012 with the industrial founding partners being United Technologies Corporation (UTC) and IBM, and the academic ones being the University of California at Berkeley, and the California Institute of Technology.

UTC is a conglomerate that deals with multi-physics systems in several vertical application domains, mainly in the aerospace and building domains. In its role in the consortium, it represents companies that host designers and builders of complex systems, which we refer to in this paper as "systems houses." IBM is a global computing infrastructure and service company that is increasingly looking at planetary scale problems. Its role in iCyPhy is that of a "technology provider," serving large systems manufacturers in automotive, aerospace, and electronics, as well as cities and nations in their attempt to optimize services such as water, energy, health, and traffic management. IBM develops systems engineering tools, with emphasis of being "the integrator" of multiple engineering disciplines, tools, and application providers. In addition, IBM is active in specific systems engineering verticals such as requirement management, architecture management, quality management and collaboration.

UC Berkeley, which leads the consortium, brings broad expertise in systems design, modeling, and analysis. Berkeley has a proven track record of changing industries through improvements in design methodologies and tools, as evidenced by its impact on electronic design automation. Caltech brings key expertise in rigorous approaches to complex, multi-physics, cyber-physical systems design and analysis, most particularly by combining the principles of control systems engineering with those of formal verification.

2 Gap Analysis

The research topics are based on an analysis of the gaps that System Houses and Technology Providers are experiencing.

2.1 System House Gap Analysis

It has always been a goal of diversified systems houses to find synergies among apparently different industrial domains. These synergies often exist at the business level, but they are more difficult to achieve in engineering. A foundational assumption of iCyPhy is that general system-level design approaches lead to substantial rationalization in design, yielding processes that are leaner and more effective while substantially reducing time-to-market by re-using components and employing correct-by-construction methods.

2.1.1 Requirements Capture, Analysis and Domain Specific Modeling

Requirements capture plays an important role in today's development processes. Requirements capture is largely natural language based and leads to many iterations

due to requirements ambiguity and lack of standard requirements libraries. The specific needs include:

- We need semi-formal and formal languages that reduce ambiguity and enable analysis for integrated systems. The level of requirements often is non-homogeneous and includes system performance requirements, safety requirements, system constraints, and customer-specific preferred solutions. The new methods should support different types of requirements to create formal (executable and analyzable) models for multiple domains.
- Because requirements are evolving throughout programs, we need analysis techniques that determine the impact of requirements changes on large interconnected systems.
- We need synthesis of early views of a system to reason about requirements validity internally, and with customers and suppliers. This requires domain-specific views of the requirements (e.g. mechanical, electrical, software, embedded hardware) at different levels of abstraction and the ability to capture cross-domain relationships.
- We need requirements modeling methods to support refinement into detailed design phases such that the design artifacts can be reused and designers can quickly iterate across abstraction levels.

2.1.2 System Integration and Views

Today's methods for system modeling are insufficient to allow for a formal model-based design flow. Most methods and tools are limited to single domains, and interconnecting these methods and tools is difficult or in some cases impossible. Existing methods and tools for cross-domain modeling and analysis lack clear semantics. More specifically:

- System modeling needs to be able to capture relationships between different domains (mechanical, electrical, software), as well as to enable analysis that crosses these domains (e.g. system reliability, performance, robustness). In particular, methods and tools are needed:
  - To reason about fault tolerance of systems and the impact of system degradation. This includes physical systems and failure modes, control system functionality, and the allocation of functionality to embedded platforms.
  - To capture and explore designs that cross multiple domains. The current practice is to limit the design space early by fixing certain decisions based on legacy knowledge and architectures and solutions from prior programs. This impedes design-space exploration and is not sufficient for programs where new architectures are introduced.
- Methodologies are needed that support the parallel nature of development programs. This requires both bottom-up and top-down capture of interfaces and constraints (e.g. through contracts).

- In addition to the integration of different views of the models, integration of different analysis methods within these views is needed. For example, the correlation between simulations and formal timing analysis is done today using independent models that have no formal relationship. Changes in architectures or requirements often make it impossible to quickly reuse such analysis.

2.1.3 Risk Management

Risk management is the identification, assessment, and prioritization of risks (defined in ISO 31000 as the effect of uncertainty on objectives), followed by coordinated and economical application of resources to minimize, monitor, and control the probability and impact of unfortunate events or to maximize the realization of opportunities [9, p. 46]. Risks can come from uncertainty in project failures (at any phase in design, development, production, or sustainment life-cycles), legal liabilities, accidents, natural causes and disasters as well as deliberate attack from an adversary, or events of uncertain or unpredictable root-cause.

Risk mitigation is a complex task. Among the risks that have to be considered in design, the ones due to uncertainties in the design parameters and in the specifications are of particular interest. The robustness of the design with respect to uncertainties in these two spaces must be addressed. Maintaining the correct operation of the system requires design centering, i.e., the choice of the nominal design variables so that the constraints are satisfied even if these variables drift from their nominal value due to the manufacturing process, product aging and adversarial environments. Sometimes centering the design is not sufficient to achieve the desired yield. In such cases, the choice and the determination of the range of tuning parameters is critical in allowing a company to adjust the design after manufacturing so that the constraints can be satisfied even though they are not satisfied in the non-tuned configuration. We are interested in design methods that allow to improve the yield of the design with respect to parametric variations and product capability of coping with an unpredictable environment.

2.2 Technology Provider Gap Analysis

Today, systems houses require engineering processes and tools that go way beyond what is known and available today. This is both a methodology and technology-intensive endeavor.

2.2.1 Formality and Usability

Historically, tools and methodologies that have had the most impact are those with strong formal foundations. Consider for example the classical engineering discipline

of feedback control, a key enabler for many engineered systems. The discipline depends on a long history of mathematical models and analytical techniques that make stability and robustness analysis possible. Consider also the field of electronics, where today's circuits have a level of complexity and reliability that was unimaginable a few short years ago. The enabling tools and methodologies in this area have solid foundations in algorithms, discrete-event systems theory, synchronous-reactive concurrency theory, and automata theory. These foundations led in the 1980s to automatic layout, logic synthesis, and RTL-based design methodologies, which were developed primarily at Berkeley and deployed in industry through close collaboration with a number of companies.

Fundamentally, formal foundations enable automated analysis and synthesis. A model that is formal is analyzable by machine, whereas an informal model is only analyzable by humans. For example, it is routine today in electronic design automation to apply formal analysis methods that effectively verify behaviors of systems over all possible input conditions (e.g. model checking).

Usable tools do exist today, but they are often based on *ad hoc* heuristics that do not offer the sorts of guarantees enabled by machine analysis. Specifically, the gaps are:

- Use of natural language in requirements capture causes inconsistency and ambiguity. The scale of contemporary projects is only aggravating the problem.
- Interfaces between subsystems lack of formal definitions. Static and dynamic properties of such interfaces are often poorly characterized. As a result, system providers cannot perform or cannot trust analysis or synthesis of systems.
- Applying formal verification in system design today usually requires construction from scratch of new models of the systems. The models used in practice by engineers are not amenable to such verification. But the construction of these new models requires deep expertise in both the application domain and in the formal modeling techniques. This combination of expertise is rare among engineers. The skill gap impedes the use of formal verification.

The need for formal techniques as a way to bridge these gaps is evident. In the context of cyber-physical systems, tools with formal foundations do exist, but, in their present form they are not usable by an average system designer. Engineers need user-friendly languages with solid formal foundations to define requirements, behaviors, constraints, and interfaces. Once they have this, with enough coverage and expressivity, they will be able to make significant advances into virtual integration and analysis as well as into generative design of complex systems.

Formal methods and tools do not take humans out of the design process. It is well known that many success stories in formal verification have arisen not from the machine analysis enabled by formal foundations, but rather from the process of constructing the new models that today's verification techniques require. The (human) process of model building exposes flaws in the design. Using languages with formal foundations favors designer intuition because models become more readily understood by the designer. The subsequent machine analysis of the models that is enabled by their formal foundations then builds confidence in the revised designs.

To favor designer intuition, tools need to integrate natural and intuitive front end languages with a formal back-end.

Formal methods and tools also do not reduce the need for simulation. On the contrary, they increase the value of simulation. When simulators are based on languages with well-understood and rigorous semantics, engineers can have confidence in the results of simulation.

Better tools and methodologies will be shorten project times, reduce redesign cycles, reduce the cost of testing, and enable better (more capable, robust, safe, and inexpensive) systems. Formal foundations will let model-based systems engineering (MBSE) live up to its full potential, and will speed its adoption.

2.2.2 Multitude of Domains

Several domains are already model driven (e.g. mechanical, electrical, and aeronautical), in the sense that engineers build and use models as an integrated part of the design process. The challenge today is the integration of diverse domains. Integration of diverse domains is intrinsic in todays complex, cyber-physical systems. It cannot be avoided. Today, this is often done locally in an ad-hoc manner, often with homegrown tools. This reduces the value of such integration, because it is hard to build confidence in the results. Or worse, integration is not done at all until late in the design process, when manufactured prototypes are put together for the first time.

In cyber-physical systems, one of the key challenges is that models must include both models of physical systems, which are rooted in continuous space and time, with models of computational systems, which are rooted is discrete, algorithmic (step-by-step) models [14]. The underlying principles of these two classes of models are well established, but many of the formal properties of the individual model evaporate when they are combined. For example, feedback control analysis in a time continuum may prove a system stable, while a software realization of the controller may yield instability. A pressing need today is to connect continuous-time physical device models with discrete models of computing systems. This integration must take as input the current industrial state-of-the-art tools and standards (e.g. SysML, Modelica, and Simulink) as entry points for such integration.

2.2.3 Complexity

The use of embedded computers and networks in complex systems today (such as cars and aircraft) has dramatically improved both the functionality and quality of these systems. These systems are more capable than before, and indeed, market pressures for ever more elaborate capability are enormous. But increased capability usually also comes with increased complexity. The key questions in product design becomes not those about affordability and return on investment, but rather about feasibility. The question is whether it is even *possible* to design the next generation products.

3 The Principles and Long-Term Focus of the Consortium

We believe the most promising means to address the challenges in systems engineering of cyber-physical systems is to employ structured and formal design methodologies that seamlessly and coherently combine the various dimensions of the multi-scale design space (be it behavior, space, or time), that provide the appropriate abstractions to manage the inherent complexity and heterogeneity, and that can provide correct-by-construction implementations.

The following issues are being addressed by the consortium:

- ***Design Methodologies***
- ***Heterogeneous Modeling and Tool Integration***
- ***Formal Control Synthesis***
- ***Design-space Exploration***
- ***Design Drivers***

We believe that it is essential to address the entire process and not to consider only point solutions of methodology, tools, and models that ease part of the design. The consortium is addressing both foundations of modeling and algorithms and solutions involving tools, flows, and methodologies. The two parts are strongly interdependent, thus forming a unified body of work that is intended to transform radically the way we do design today.

This research requires novel principles. Academic research in methodology and design frameworks has been successful in the electronics domain because of the early involvement of industry in driving the needs and testing the results. In the case of systems engineering, academic work has not yet had the industrial impact that many hoped for. Given the size and complexity of the problems, a significant number of researchers must be involved. For this reason, the consortium is structured as a close collaboration between industry and academia. The research is driven by industrial needs, but cannot ignore foundational issues.

3.1 Research Summary

The taxonomy shown in Figure 1 gives a perspective on high-level research topics, how these research topics link together, and how they address specific industrial concerns (the light bubbles). The adjacencies can be exploited to derive solutions that are more performing than they would if developed in isolation. The scope of research is large. Foundational work is pervasive and relies upon an overall emphasis on formal methods and a rigorous approach to design.

The topics are briefly described below and are grouped according to the diagram shown in Figure 1. In each topic we will underline the foundation and the solution components.

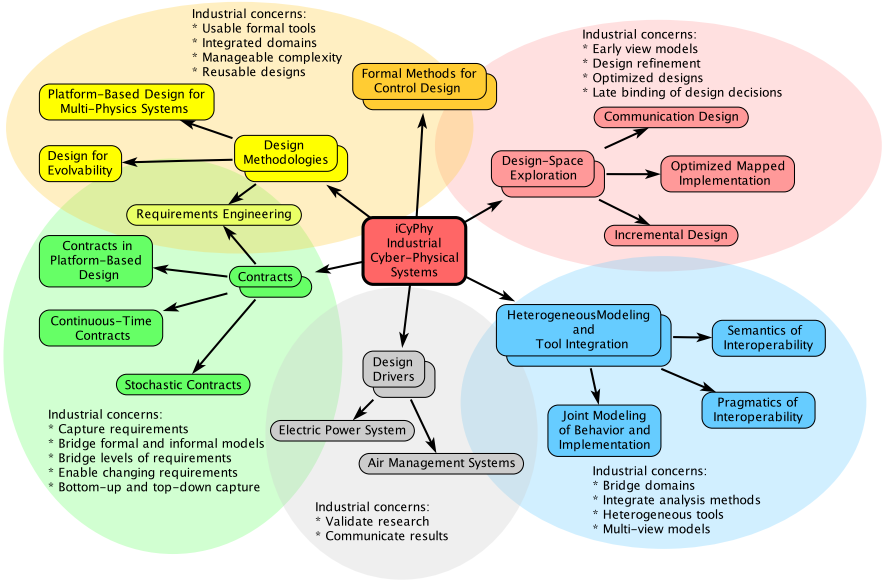

Fig. 1 Taxonomy of research needs.

3.2 Design Methodologies

This section addresses model-based systems engineering for cyber-physical systems.

3.2.1 Platform-Based Design for Multi-Physics Systems

Driven by the industrial needs to integrate different physical domains, and by the diversity of modeling and simulation tools and methods for these physical domains, our focus is on principles of composability, abstraction, and refinement. Recent progress in object-oriented modeling languages like Modelica [6] suggests directions for such principles that can embrace mechanical, electrical, and thermal domains, for example. Key challenges include how to define precisely the notion of composability of components, and how to characterize the interfaces of components so that they can be composed in a natural and correct-by-construction fashion. Further, the notions of abstraction and refinement need to be developed, possibly relating these concepts to the mathematical approach to reduced-order modeling. The analysis and verification of components and of integrated systems need to leverage best-of-class tools that deal with different physical phenomena. For example, to analyze structural properties together with thermal behavior of a system may require composing significantly different tools and methodologies. How to relate the var-

ious domains so that they maintain consistency across layers of abstraction is an open problem, especially when the hierarchy may be different across physical domains. The composition rules and the relations among different viewpoints will be captured using contracts (see below).

3.2.2 Design for Evolvability

We are working to endow modeling languages with semantics that adequately express required behavior in a technology-independent way, and to provide synthesis tools that yield a multiplicity of implementations that, by construction, have the behavior specified in the models. Such semantics must include an ability to express temporal behavior and temporal requirements independently from the underlying system implementation. In the near term, we will evaluate the extent to which controlling timing in software can lead to designs that are more robust to changes in the implementation platform [15]. We will also evaluate the effectiveness of representations of temporal behavior in existing industrial modeling languages and tools such as AADL, Simulink, and SysML, and research modeling languages and tools such as Giotto [8] and PTIDES [4]. Finally, we are developing a simulation platform that enables comparison of behaviors of models executing on a variety of implementation platforms.

3.2.3 Requirements Engineering

Requirements capture plays a major role in industrial design processes. Our key goal is a more formal approach that will enable automated analysis. This includes methods for specifying constraints on timing, dynamic behavior, and static properties.

3.2.4 Contract-based Design

Driven by the needs of industry to introduce formality in the design process, iCyPhy is focusing on contracts as a formalization of the rules for composition, abstraction, and refinement. Making contract-based design a technique of choice for system engineers, the team is developing mathematical foundations for contract representation and requirement engineering that enable the design of frameworks and tools. For example, a type-theoretic system for building domain-specific ontologies and annotating components with constraints on such ontologies enables better compatibility checking between separately developed components [18].

Contracts in Platform-Based Design. To integrate methodologies and tools, iCyPhy is merging contract-based design with platform-based design [24] to formulate the design process as a meet-in-the-middle approach, where design requirements are implemented in a subsequent refinement process using as much as possible elements

from a library of available components. Contracts are formalizations of the conditions for correctness of element integration (horizontal contracts), for a lower level of abstraction to be consistent with the higher ones, and for abstractions of available components to be faithful representations of the actual parts (vertical contracts).

A typical use of contracts in cyber-physical system design would be to govern the horizontal composition of the cyber and the physical components and to establish the conditions for correctness of their composition.

Continuous-Time Contracts. To use contracts for heterogeneous domains, we need a theory for continuous-time contracts. Effective continuous-time contracts require (i) models of time that can semantically distinguish discrete events and continuous change, (ii) expressing bounds on continuous behaviors in terms of discrete constraints, and (iii) provide for interaction between simulation models and models for semantic analysis and design. In fact, a full-fledged continuous-time contract theory should support dynamical models as well as structural and performance models so as to be practical for CPS design. The long-term goal is to provide the conceptual framework for such contracts and software prototypes that demonstrate its efficacy.

In the near term, the plan is to extend techniques that have shown promise in the context of mixed-signal (analog/digital) integrated circuit design [22] to encapsulate the physical portion of the CPS and provide a "generalized" interface to its "cyber" counterpart. Such a generalized interface should offer:

- Composition constraints and rules for the interaction of a subsystem with its environment; such constraints are formalized with horizontal contracts;
- Simplified discrete-time and amplitude-quantized behavioral models for efficient design exploration and co-simulation of a subsystem with its environment; the range of validity of behavioral models are defined by bottom-up vertical contracts;
- Information about the capabilities of the subsystems in terms of timing, power consumption, size, weight and other physical aspects (performance models) that need to be transmitted to the system assemblers to allow for early detection of design errors; the range of usage of such performance models is defined by top-down vertical contracts.

Stochastic Contracts. To address the industrial needs of fault tolerance and robustness of design, we need to consider stochastic contracts. Complex systems are stochastic in nature. In fact, several parameters impacting both the behavior and the performance of these systems are subject to variability due to manufacturing tolerances, usage, and faults. Moreover, models and abstractions that are normally used to design multi-physics systems inevitably introduce inaccuracies, since either the dynamics of the several components are not perfectly known, or approximations are needed to guarantee efficient explorations. As a consequence, robust system design can often imply costly characterization based on Monte Carlo simulations or expensive overdesign to guarantee large safety margins.

The long-term objective is to provide a conceptual framework for such contracts to support analysis and design stochastic heterogeneous systems, and the software prototypes that demonstrate its efficacy.

In the near term, iCyPhy is focusing on the broad set of existing stochastic models that can capture both the continuous and discrete dynamics of cyber-physical systems, such as Markov jumps linear systems, piecewise deterministic Markov processes, stochastic hybrid systems, and switching diffusion processes, on which contracts need to be formulated. Because of the heterogeneity of stochastic hybrid systems, several models can indeed be adopted, depending on which dynamics are affected by uncertainties.

3.3 Heterogeneous Modeling and Tool Integration

The challenge is to define models of computation (MoCs) that are sufficiently expressive and have strong formal properties that enable systematic validation of designs and correct-by-construction synthesis of implementations. A second challenge is to identify which of the many MoCs and variants are actually needed, to figure out how to educate the community to use them, and to articulate the choices into industrial standards such as SysML. The major innovation being pursued in iCyPhy concerns the interoperability of MoCs, thereby enabling heterogenous design with rigorous foundations.

Semantics of Interoperability. MoCs are built by combining three largely orthogonal aspects: sequential behavior, concurrency, and communication. Similar to the way that an MoC abstracts a class of behavior, "abstract semantics" abstract the semantics of the MoC itself [16]. The concept is called a "semantics meta-model" in [25], but since the term "meta-model" is more widely used in software engineering to refer instead to models of the structure of models (see [21] and `http://www.omg.org/mof/`), we prefer to use the term "abstract semantics" here. The concept of abstract semantics is leveraged in Ptolemy II [5] and Metropolis [1] to achieve heterogeneous mixtures of MoCs with well-defined interactions.

The key challenge is providing actor-oriented MoCs [17] with well-defined semantics. All too often, the semantics emerge accidentally from the software implementation rather than being built in from the start. One of the key challenges is to integrate actor-oriented models with practical and realistic notions of time. To address, for example, modeling distributed behaviors, it is essential to provide multiform models of time. Modeling frameworks that include a semantic notion of time, such as Simulink and Modelica, assume that time is homogeneous in the sense that it advances uniformly across the entire system. In practical distributed systems, even those as small as systems-on-chip, however, no such homogeneous notion of time is measurable or observable. In a distributed system, even when using network time synchronization protocols (such as IEEE 1588 [10]), local notions of time will differ, and failing to model such differences could introduce artifacts in the design.

Pragmatics of Interoperability. Despite considerable progress in languages, notations, and tools, major problems persist. In practice, system integration, adaptation of existing designs, and interoperation of heterogeneous subsystems remain major stumbling blocks that cause project failures. We believe that model-based design, as widely practiced today, largely fails to benefit from the principles of platform-based design [24] as a consequence of its lack of attention to the semantics of heterogeneous subsystem composition.

Many previous efforts have focused on tool integration, where tools from multiple vendors are made to interoperate [19, 7, 11]. This approach is challenging, however, and yields fragile tool chains. Many tools do not have adequate published extension points, and maintaining such integration requires considerable effort.

iCyPhy believes a better approach is to focus on the semantics of interoperation, rather than the software problems of tool integration.

Nevertheless, a purely semantics-based approach will fail to have practical impact in industry because it will not embrace industry-standard tools. A promising recent development is the evolving Functional Mockup Interface (FMI) standard (see `https://fmi-standard.org`), which aims to enable model exchange and co-simulation between continuous-time models. The iCyPhy consortium is actively involved in the development of this standard with the goal of ensuring that it is capable of support a sound semantics of interoperation.

Joint Modeling of Behavior and Implementation. The Metropolis project [1, 3] has introduced the notion of a "quantity manager," a component of a model that functions as a gateway to another model. For example, a purely functional model that describes only idealized behavioral properties of a flight control system could be endowed with a quantity manager that binds that functional model to a model of a distributed hardware architecture using a particular network fabric. By binding these two models, designers can evaluate how properties of the hardware implementation affect the functional behavior of the system. The iCyPhy consortium is further developing this concept, generalizing it as a form of aspect-oriented modeling [12] and integrating it with the Ptolemy framework.

3.4 Formal Methods for Control Design

The co-design of controllers and certificates of correctness is emerging as a promising approach for "correct by construction" design that addresses issues in verification and integration [2, 13, 28]. There are a number of broad research directions available based on the initial work we have done in this area.

Performance specifications. Current techniques for control protocol synthesis often provide correct behavior but with no regard for performance. It will be important to add in the ability to include cost (or reward) functions in temporal logic planning, allowing protocols that satisfy a set of (hard) constraints as well as minimizing a cost function associated with the continuous or discrete states [27]. A related issue is al-

lowing optimization of the probability that certain specifications are met, to move away from pure worst-case performance [26]. Including the ability to specify real-time properties (such as the amount of time between an environmental event and the response of the control system) is also an area in where performance specifications must be generalized. This might build on work in the computer science literature on timed automata and real-time temporal logics, but also incorporate continuous dynamics and control actions [29].

Controller architecture. Most existing techniques focus on the synthesis of a single, centralized controller. It will be important to develop methods for designing and validating formal interface specifications between subsystems (horizontal contracts) that allows verification and synthesis to be performed at the subsystem level, with guaranteed system level requirements [23]. We must also develop hierarchical control structures that make use of a demand-response architecture and formal interface specifications between layers (vertical contracts) to achieve a system-level goal. Preliminary work in control of autonomous vehicles provides a starting point for this work [28].

Over the long term, we seek to derive and implement algorithms for synthesis of control protocols that can be applied to cyber-physical systems. Key elements are increasing our ability to capture dynamics, uncertainty and feedback in our theory and integrating new algorithms for solving the types of problems identified above into the TuLiP (or other) open-source software packages.

3.5 Design Space Exploration

Communication Design. System designers today are leveraging as much as possible computing and networking technology to gain capability and performance and to reduce costs. One such approach is to select Ethernet as the physical layer for communication in embedded systems such as airplanes and cars. Since the protocol used with Ethernet is asynchronous, there are serious concerns about the safety implications of this choice. A solution to this problem is to use an additional protocol layer on top of Ethernet that would provide a synchronous platform for the applications. TTEthernet, Arinc 429/717, and Audio Video Bridging (AVB) Ethernet are all defining synchronous (time triggered) Ethernet-based standards. One approach being pursued in iCyPhy is PTIDES, which leverages these networking developments to provide a foundation for distributed software with controllable timing properties [4].

Optimized Mapped Implementation. If we want to perform automatic optimal mapping of behaviors onto architectural elements, we must embed behavior and architecture in the same semantic domain. For example, in automatic logic synthesis register-transfer level (RTL) descriptions and gate representations are mapped into a particular form of Boolean representation called the Boolean network. Doing this,

we can optimize the mapping process by using a covering algorithm. We maintain that this process is indeed applicable to all layers of abstraction provided that a common semantic domain that makes the mapping algorithm effective is found.

Incremental Design. This research topic is related to the need to understand the impact of design changes on the performance, cost, and time to market. Albeit using formal languages and synthesis has been a key methodology approach in moving VLSI design to a level of productivity that was unimaginable before this approach was introduced, it did expose design implementations to instability with respect to changes in the sense that a small change in behavior can result in a large, unpredictable variation in the logic implementation. This yielded a large amount of redesign in subsequent implementation steps and in particular, in the layout of the integrated circuit under design that created havoc with schedule and chip size estimations.

This effect was the result of the optimization process that is notoriously unstable with respect to input changes if not constrained. A method that was developed in the VLSI domain was to limit the re-design due to the changes to a part of the previous implementation. Which part to choose was the research problem to be faced and that was only partially resolved in that application domain.

The consortium is studying the stability problem in design space exploration by limiting the degrees of freedom that are used in the optimization steps and in particular, the allocation of functionalities to architectural blocks.

3.6 Design Drivers

An effective industry-academic collaboration leverages the real-world systems experience of industry to test and refine academic models and tools. ICyPhy has focused on two richly heterogeneous system problems, namely the electric power and air management systems (EPS and AMS) of advanced aircraft.

Electric Power Systems. The EPS is a key subsystem of an aircraft vehicle management systems (VMS) [20]. Its function is to generate, regulate, and distribute electrical power throughout the aircraft.

EPS design poses several challenges, including controller architecture definition, contactor and sensor optimization, safety and fault tolerance, and efficient load management. Given an EPS topology, typically captured by a so-called single-line diagram (SLD), there is clearly no unique solution for the bus power control unit (BPCU), since both the controller inputs (sensor number and location) and outputs (contactor number and location), essential elements for controller design, are underspecified and left as design choices. The initial SLD structure itself may not guarantee the desired reliability level and may require modifications in terms of component and path redundancy.

An EPS system also has multi-physics aspects. The dynamics of generators and loads can get quite complex. The mechanical parts also affect system behav-

ior, where latency and bounce in contactors can affect behavior. An it has cyber-physical aspects, since EPS systems today are implemented using networked microcontrollers.

Nearly every thrust within iCyPhy can be applied to an EPS design. Given a set of loads, the power system can be built out of a library including, among other components, generators, buses, power converters, sensors and contactors. System requirements are expressed in terms of safety, reliability, and availability constraints. EPS design is framed as an optimization problem where the selected candidate topologies and controller architectures satisfy all the requirements, while optimizing quality factors such as weight, efficiency, complexity, and cost. To achieve correct-by-construction design, we formulate contracts at different articulation points in the design flow. Controller synthesis techniques from Section 3.4 ensure realization of logical specifications. Horizontal contracts will formalize the conditions under which component integration is correct; vertical contracts will formalize the conditions under which an implementation is consistent with its abstraction, or an abstraction is a faithful representation of an implementation. Co-simulation of functionality and architecture enables studying how, for example, network architecture affects dynamics. Multi-physics simulation enables analysis of how network dynamics affects electrical dynamics. Finally, stochastic contracts can establish conditions under which safety and reliability requirements are guaranteed.

Air Management Systems. A second application driver is the air management system of an aircraft. This application has more diverse multi-physics aspects, since thermodynamics, fluid dynamics, and mechanical geometry all come into play. A significant challenge is to develop techniques that enable use of best-of-class industry-standard modeling and simulation tools together with the new tools and methodologies being developed.

4 Conclusion

Progress in systems engineering for cyber-physical systems requires a deep collaboration between industry and academia. The iCyPhy industry-academic partnership has been formed to develop a new generation of system modeling, design, and analysis tools and methodologies that will enable more effective design of more capable systems.

5 Acknowlegements

Thanks to John Arnold for helping to form the vision in this paper.

References

1. F. Balarin, H. Hsieh, L. Lavagno, C. Passerone, A. L. Sangiovanni-Vincentelli, and Y. Watanabe. Metropolis: an integrated electronic system design environment. *Computer*, 36(4), 2003.
2. C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas. Symbolic planning and control of robot motion [Grand Challenges of Robotics]. *Robotics & Automation Magazine, IEEE*, 14(1):61–70, Mar. 2007.
3. A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, and Q. Zhu. A next-generation design framework for platform-based design. In *Design Verification Conference (DVCon)*, San Jose', California, 2007.
4. J. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou. Distributed real-time software for cyber-physical systems. *Proceedings of the IEEE (special issue on CPS)*, 100(1):45–59, 2012.
5. J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity—the Ptolemy approach. *Proceedings of the IEEE*, 91(2):127–144, 2003.
6. P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Wiley, 2003.
7. Z. Gu, S. Wang, S. Kodase, and K. G. Shin. An end-to-end tool chain for multi-view modeling and analysis of avionics mission computing software. In *Real-Time Systems Symposium (RTSS)*, pages 78 – 81, 2003.
8. T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A time-triggered language for embedded programming. *Proceedings of IEEE*, 91(1):84–99, 2003.
9. D. Hubbard. *The Failure of Risk Management: Why It's Broken and How to Fix It*. John Wiley & Sons, 2009.
10. IEEE Instrumentation and Measurement Society. 1588: IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. Standard specification, IEEE, November 8 2002.
11. G. Karsai, A. Lang, and S. Neema. Design patterns for open tool integration. *Software and Systems Modeling*, 4(2):157–170, 2005.
12. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J. M. Loingtier, and J. Irwin. Aspect-oriented programming. In *ECOOP, European Conference in Object-Oriented Programming*, volume LNCS 1241, Finland, 1997. Springer-Verlag.
13. H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu. Correct, Reactive, High-Level Robot Control. *Robotics & Automation Magazine, IEEE*, 18(3):65–74, 2011.
14. E. A. Lee. Cyber physical systems: Design challenges. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC),*, pages 363 – 369, Orlando, Florida, 2008. IEEE.
15. E. A. Lee. Computing needs time. *Communications of the ACM*, 52(5):70–79, 2009.
16. E. A. Lee. Disciplined heterogeneous modeling. In D. C. Petriu, N. Rouquette, and O. Haugen, editors, *Model Driven Engineering, Languages, and Systems (MODELS)*, pages 273–287. IEEE, 2010.
17. E. A. Lee, S. Neuendorffer, and M. J. Wirthlin. Actor-oriented design of embedded hardware and software systems. *Journal of Circuits, Systems, and Computers*, 12(3):231–260, 2003.
18. B. Lickly, C. Shelton, E. Latronico, and E. A. Lee. A practical ontology framework for static model analysis. In *International Conference on Embedded Software (EMSOFT)*, pages 23–32. ACM, 2011.
19. J. Liu, B. Wu, X. Liu, and E. A. Lee. Interoperation of heterogeneous CAD tools in Ptolemy II. In *Symposium on Design, Test, and Microfabrication of MEMS/MOEMS*, Paris, France, 1999.
20. I. Moir and A. Seabridge. *Aircraft Systems: Mechanical, Electrical, and Avionics Subsystems Integration*. AIAA Education Series. Wiley, third edition edition, 2008.
21. G. Nordstrom, J. Sztipanovits, G. Karsai, and A. Ledeczi. Metamodeling - rapid design and evolution of domain-specific modeling environments. In *Proc. of Conf. on Engineering of Computer Based Systems (ECBS)*, pages 68–74, Nashville, Tennessee, 1999.

22. P. Nuzzo and A. Sangiovanni-Vincentelli. Robustness in analog systems: Design techniques, methodologies and tools. In *Proc. IEEE Symp. Industrial Embedded Systems*, Jun. 2011.
23. N. Ozay, U. Topcu, and R. M. Murray. Distributed power allocation for vehicle management systems. In *Proc. IEEE Control and Decision Conference*, 2011.
24. A. Sangiovanni-Vincentelli. Defining platform-based design. *EEDesign of EETimes*, 2002.
25. A. Sangiovanni-Vincentelli, G. Yang, S. K. Shukla, D. A. Mathaikutty, and J. Sztipanovits. Metamodeling: An emerging representation paradigm for system-level design. *IEEE Design and Test of Computers*, 2009.
26. E. M. Wolff, U. Topcu, and R. M. Murray. Robust control of uncertain markov decision processes with temporal logic specifications. In *Proc. IEEE Control and Decision Conference*, 2012.
27. E. M. Wolff, U. Topcu, and R. M. Murray. Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic. In *Proc. IEEE Control and Decision Conference*, 2013.
28. T. Wongpiromsarn, U. Topcu, and R. M. Murray. Synthesis of control protocols for autonomous systems. *Unmanned Systems*, 1(1):21–40, 2013.
29. H. Xu. *Design, specification, and synthesis of aircraft electric power systems control logic*. PhD thesis, California Institute of Technology, 2013.