THE INSTITUTE FOR SYSTEMS RESEARCH

ISR TECHNICAL REPORT 2014-02

Development of a Framework for CPS Open Standards and Platforms

John S. Baras and Mark A. Austin

TECHNICAL REPORT

Systems Engineering and Integration Laboratory (SEIL)
Institute for Systems Research
University of Maryland
College Park, MD 20742, USA

DEVELOPMENT OF A FRAMEWORK FOR CPS OPEN STANDARDS AND PLATFORMS

by

John S. Baras and Mark A. Austin

November 8, 2013

Acknowledgment: Research supported in part by Cooperative Agreement, NIST 70NANB11H148, to the University of Maryland College Park.

DEVELOPMENT OF A FRAMEWORK FOR CPS OPEN STANDARDS AND PLATFORMS

1. Introduction

This technical report describes a Framework we have developed through our research and investigations in this project, with the goal to facilitate creation of Open Standards and Platforms for CPS; a task that addresses a critical mission for NIST.

The rapid development of information technology (in terms of processing power, embedded hardware and software systems, comprehensive IT management systems, networking and Internet growth, system design environments) is producing an increasing number of applications and opening new doors. In addition over the last decade we entered a new era where systems complexity has increased dramatically. Complexity is increased both by the number of components that are included in each system as well as by the dependencies between those components. Increasingly, systems tend to be more software dependent and that is a major challenge that engineers involved in the development of such systems, face. The challenge is even greater when a safety critical system is considered, like an airplane or a passenger car. Software-intensive systems and devices have become everyday consumables. There is a need for development of software that is provably error-free. Thanks to their multifaceted support for networking and inclusion of data and services from global networks, systems are evolving to form integrated, overarching solutions that are increasingly penetrating all areas of life and work. When software dependent systems interact with the physical environment then we have the class of cyber-physical systems (CPS) [1, 2]. The challenge in CPS is to incorporate the inputs (and their characteristics and constraints) from the physical components in the logic of the cyber components (hardware and software). CPS are *engineered systems* constructed as networked interactions of physical and computational (cyber) components. In CPS, computations and communication are deeply embedded in and interacting with physical processes, and add new capabilities to physical systems. Competitive pressure and societal needs drive industry to design and deploy airplanes and cars that are more energy efficient and safe, medical devices and systems that are more dependable, defense systems that are more autonomous and secure. Whole industrial sectors are transformed by new product lines that are CPS-based.

Modern CPSs are not simply the connection of two different kinds of components engineered by means of distinct design technology, but rather, a new system category that is both physical and computational [1, 2]. Current industrial experience tells us that, in fact, we have reached the limits of our knowledge of how to combine computers and physical systems. The shortcomings range from technical limitations in the foundations of cyber-physical systems to the way we organize our industries and educate engineers and scientists that support cyber-physical system design. If we continue to build systems using our very limited methods and tools but lack the science and technology foundations, we will create significant risks, produce failures and lead to loss of market.

Nowadays, with increasing frequency we observe systems that cooperate to achieve a common goal, even though there were not built for that reason. These are called *systems of systems*. For example, the Global Positioning System (GPS) is a system by itself. However, it needs to cooperate with other systems when the air traffic control system of systems is under consideration. The analysis and development of such systems should be done carefully because of the emergent behavior that systems exhibit when they are coupled with other systems. However, apart from the increasing complexity and the other technical challenges, there is a need to decrease time-to-market for new systems as well as the associated costs. This specific trend and associated requirements, which are an outcome of global competitiveness, are expected to continue and become even more stringent.

If a successful contribution is to be made in shaping this change, the revolutionary potential of CPS must be recognized and incorporated into internal development processes at an early stage. For that *Interoperability and Integratability of CPS* is critical. In this Task we have developed a Framework to facilitate interoperability and integratability of CPS via *Open Standards and Platforms*. The purpose of this technical report is to introduce this Framework and its critical components, to provide various instantiations of it, and to describe initial successful applications of it in various important classes of CPS. An additional goal of publishing this technical report is to solicit feedback on the proposed Framework, and to catalyze discussions and interactions in the broader CPS technical community towards improving and strengthening this Framework.

CPS integrate data and services from different systems which were developed independently and with disparate objectives, thereby enabling new functionalities and benefits. Currently there is a lack of well-defined interfaces that on the one hand define the standards for the form and content of the data being exchanged, but on the other hand take account of non-functional aspects of this data, such as differing levels of data quality or reliability. A similar situation exists with respect to tools and synthesis environments, although some work has been initiated in the latter.

The technological prerequisite for the design of the aforementioned various functions and value added services of CPS is the interoperability and integratability of these systems as well as their capability to be adapted flexibly and application-specifically as well as extended at the different levels of abstraction. Dependent on the objective and scope of the application, it may be necessary to integrate component functions (Embedded Systems (ES), System of Systems (SoS), CPS), to establish communication and interfaces, and to ensure the required level of quality of interaction and also of the overall system behavior. This requires cross-domain concepts for architecture, communication and compatibility at all levels. The effects of these factors on existing or yet undeveloped systems and architectures represent a major challenge. Investigation into these factors is the objective of current national and international studies and research projects.

CPS create core technological challenges for traditional system architectures, especially because of their high degree of connectivity. This is because CPS are not constructed for one specific purpose or function, but rather are open for many different services and processes, and must therefore be adaptable. In view of their evolutionary nature, they are only controllable to a limited extent. This creates new demands for greater interoperability and communication within CPS that cannot be met by current closed systems. In particular, the differences in the characteristics of embedded systems in relation to IT systems and services and data in networks lead to outstanding questions in relation to the form of architectures, the definition of system and communication interfaces and requirements for underlying CPS platforms with basic services and parallel architectures at different levels of abstraction.

The technological developments underlying CPS evolution require the development of standards in the individual application domains, as well as basic infrastructure investments that cannot be borne by individual companies alone. This is particularly significant for SMEs. The development and operation of uniform platforms to migrate individual services and products will therefore be as much of a challenge as joint specification standards. The creation of such quasi standards, less in the traditional mold of classic industry norms and standards and more in the sense of de facto standards that become established on the basis of technological and market dominance, will become an essential part of technological and market leadership.

To summarize and emphasize, the complexity of the subject in terms of the required technologies and capabilities of CPS, as well as the capabilities and competences required to develop, control and design/ create innovative, usable CPS applications, demand fundamentally integrated action, interdisciplinarity (research and development, economy and society) and vertical and horizontal efforts in:

- The creation of open, cross-domain platforms with fundamental services (communication, networking, interoperability) and architectures (including domain-specific architectures);
- The complementary expansion and integration of application fields and environments with vertical experimentation platforms and correspondingly integrated interdisciplinary efforts;
- The systematic enhancement with respect to methods and technologies across all involved disciplines to create innovative CPS.

The aim of our research and investigations under this Task of the project, was precisely to clarify these objectives and systematically develop detailed recommendations for action. Our research and investigations have identified the following essential and fundamental challenges for the modeling, design, synthesis and manufacturing of CPS:

(i) The creation and demonstration of a framework for developing cross-domain integrated modeling hubs for CPS.

(ii) The creation and demonstration of a framework for linking the integrated CPS modeling hub of (i) with powerful and diverse tradeoff analysis methods and tools for design exploration for CPS.

(iii) The creation of a framework of linking the integrated CPS synthesis environment of (i) and (ii) with databases of modular component and process (manufacturing) models, backwards compatible with earlier legacy systems;

(iv) The creation of a framework for translating textual requirements to mathematical representations as constraints, rules and metrics involving both logical and numerical variables and the automatic (at least to 75%) allocation of the resulting specifications to components of the CPS and of processes, in a way that allows traceability.

These challenges have been listed here in the order of increasing difficulty both conceptually and in terms of arriving at implementable solutions. The order also reflects the extent to which the current state of affairs has made progress towards developing at least some initial instantiations of the desired frameworks. In this context, it is useful to compare with the advanced state of development of similar frameworks and their instantiations for synthesis and manufacturing of complex microelectronic VLSI chips including distributed ones, which have been available as integrated tools by several vendors for at least a decade.

Regarding challenge (i) we have performed extensive work and research in this project towards developing model-based systems engineering (MBSE) procedures for the design, integration, testing and operational management of cyber-physical systems, that is, physical systems with cyber potentially embedded in every physical component. Thus in the Framework, described in this report, for standards for integrated modeling hubs for CPS, MBSE methods and tools are prominent. Regarding the search for a framework for standards for CPS this selection has the additional advantage that it is also emerging as an accepted framework for systems engineering by all industry sectors with substantial interest in CPS [3, 7].

Regarding challenge (ii) we have performed extensive work and research in this project towards developing the foundations for such an integration, and we have developed and demonstrated the first ever integration of a powerful tradeoff analysis tool (and methodology) with our SysML-Integrated system modeling environments for CPS synthesis [3, 7]. Primary applications of interest that we have instantiated this framework are: microgrids and power grids, wireless sensor networks (WSN) and applications to Smart Grid, energy efficient buildings, microrobotics and collaborative robotics, and the overarching (for all these applications) security and trust issues including our pioneering and innovative work on compositional security systems. A key concept here is the integration of multi-criteria, multi constraint optimization with constrained based reasoning.

Regarding challenge (iii) we have only developed the conceptual Framework, as any required instantiations will require substantial commercial grade software development beyond the scope of this project. It is clear however that object-relational databases and database mediators (for both data and semantics) will have to be employed.

Regarding challenge (iv) we have developed a Framework for checking and validating specifications, after they have been translated to their mathematical representations as constraints and metrics with logical and numerical variables. Various multi-criteria optimization, constrained based reasoning, model checking and automatic theorem proving tools will have to be combined. The automatic annotation of the system blocks with requirements and parameter specifications remains an open challenge.

2. CPS Architectures

A key concept we investigated is that of architectures for CPS. This still remains a challenge. We (the MS researchers) participated in the studies of this subject as members of the NIST CPS Architecture Task Group.

Generically the ***Architecture of a System*** consists of:

(1) The arrangement of entities that constitute the system;

(2) The relationships between these entities.

For a physical system, the architecture defines the form (structure) that performs the function (behavior). It also defines the interconnections between components and the associated interfaces. For CPS, since various physics are involved in the physical components understanding and modeling such interconnections and interfaces could be very complex. The same can be said about software systems, even though the 'form/structure' descriptor does not have any geometric or material meaning as it does for the physical components. And finally the architecture of a CPS must describe the interfaces between the cyber and the physical components at different scales. Thus describing and modeling CPS architectures can be a pretty challenging task. One of the great difficulties involved is to find appropriate models and representations so that design and manufacturing engineers can explore various architectures in a systematic and quantitative manner. It is our assessment that we are not close in developing satisfactory such models and representations of CPS architectures given the current state of affairs in CPS. Neither we are close in developing a satisfactory taxonomy of architecture classes for CPS in various areas of interest. We discuss below various issues and concepts related to CPS architectures from [2, 3, 4, 5].

First, every system has at least one architecture, whether it is stated explicitly or not. In fact, a typical system has more than one architecture depending on the intended purpose for describing it. The architecture of a system is essential to:

- Understand, model and analyze complex systems;
- Design complex systems;
- Manufacture complex systems;
- Evaluate the cost and other financial concerns about a system and its potential markets;
- Design standards and protocols to guide the evolution of long-lived systems;
- Manage complex systems.

These are especially true of cyber-physical systems, where the physical and cyber (e.g., computation, communication, control) components have their own architecture(s).

One common classification of architecture of a physical system is whether it is modular or integral. Most physical system architectures lie somewhere in between. Recent scholarship [3] seems to indicate that there is a scientific limit to the modularity of physical systems architecture, even when it is desired. No such limit seems to exist for the modularity of cyber systems architecture.

While cyber system architectures can also be classified along the modular-integral axis, more frequently they are described as more/less 'layered' or 'hierarchical'. For example the popular Internet-based communication architecture consists of layers of protocols. Much of the success of the Internet is attributed to its layered architecture. For example, a typical web browser can be implemented using the protocols HTTP/TCP/IP/Ethernet, layering from top to bottom. A rich set of choices provided in each layer and the standardized interface between the layers have contributed to the success of the Internet-based communication.

Drawing from this Internet inspiration, Service-Oriented Architecture (SOA) for large scale enterprise-wide computing has adopted a layered approach. Casting computations as services is a popular trend for all sorts of computations, and these computations are architected as computing service layers with standardized interfaces.

In the world of control systems, a popular architecture is hierarchical. In this hierarchical structure each node operates independently, performing tasks received from its superior node, commanding tasks of its subordinate nodes, sending abstracted sensations to its superior node, and receiving sensations from its subordinate nodes. The leaf notes are physical sensors and actuators. Note that each level in the hierarchy can be treated as a layer, and communications between them can be architected as discussed above. A more decentralized architecture is used in distributed and networked control systems. A typical large-scale control system usually has a hybrid architecture consisting of distributed and hierarchical clusters. In addition, all control systems are evolving into a rich collection of computation and communication subsystems – thus inheriting the architectures of these subsystems.

In summary, layered architecture is emerging as a popular choice for cyber (e.g., computation, communication, control) systems, with rich choices within each layer and standardized interface between layers.

To further investigate the interesting topic of architectures for cyber-physical systems (CPS) we helped design and participated in survey of NIST subject matter experts in different domains of CPS including: SmartGrid & Telecommunication, SmartGrid, Smart Transportation – Operations, Information Technology, Building Systems, Smart Manufacturing, Wireless Emergency Networks, Health IT, General CPS. The initial responses have been collected in an informal initial report [4].

To further investigate this key topic, an invited panel on CPS Architectures organized by PI Prof. Baras at the International Conference on CPS (ICCPS) held in Philadelphia April 9-12 [7]. The invited and distinguished panelists were: Prof. Manfred Broy of Technical University of Munich (Germany), Prof. Karl-Henrik Johansson of the Royal Institute of Technology (Sweden), Dr. David Corman of Boeing (and NSF), Dr. Vijay Srinivasan of NIST, Prof. Janos Sallai of Vanderbilt University and Prof. Raj Rajkumar of Carnegie Mellon University.

The theme of this panel was to discuss concepts paradigms and needs towards developing a systematic and rigorous methodology, models and analysis for CPS architectures. Architecture is a key ingredient of any system. In a generic sense one understands by Architecture a description of the various structure and behavior components of a system together with their configuration and interfaces and interconnections. The concept of Architecture for CPS is a challenging concept as it needs to account for both the physical and cyber constraints. For instance physical and material laws as well as geometric laws and reasoning will guide the physical part. The same is true for various concepts of time and their constraints. Extensions of current distributed architectures for computers at all scales, and including both digital and analog components need to be considered. Even more importantly the interplay between the principles and rules of architectures from the physical and cyber sides need to be considered and brought to harmony.

The purpose of this panel was to initiate extensive discussions within the technical community of CPS at large with the goal to start developing principles, languages and a taxonomy of such architectures for CPS. An important new concept that was brought to the forefront was the significance of geometry and matter, which has not so far been considered in discussions of CPS architectures.

The operational scenario of the panel was as follows. Professor Baras provided a brief introduction to the topic. He also introduced the following key questions that were addressed by the panelists and the audience.

1) Examples of physical system architectures strongly influenced by the physical laws of the components, including material and geometry laws and principles.
2) Examples of system cyber architectures where the physical layer and heterogeneous engineering components played a critical role.
3) Do we need specific architecture description languages for CPS?
4) What is the current state of the art in industry sectors like automotive, aerospace, power grids, where CPS thinking has already started?
5) Visions about some generic architectures set-up like the various planes in complex communication and computer networks. Is such a generic framework appropriate or even feasible for CPS? Is it possible to develop a taxonomy of CPS architectures? Examples?
6) There are pervasive cross-cutting concerns across classes of CPS, like security-resilience and robustness. How should these requirements be reflected in CPS architectures?
7) Is there a need for standards development as we work towards a taxonomy of CPS architectures? How important are such developments for interoperability and design of CPS?
8) What should the role and principles of CPS architectures with respect to validation and verification at the system level?
9) What is the role and principles for CPS architectures form the perspective of composability and compositionality?
10) CPS exist at various scales from macro to nano and even at multiple scales within the same system. What are the challenges for CPS architectures emanating from this multi-scale reality?

Each of the panelists made a short presentation of about 10 minutes. These presentations were followed by a lively discussion and questions and answers with the audience and the panelists. The audience attendance was about 80 people. Feedback from the participants was that the panel and discussions were very interesting and timely. The panel and discussion duration was approximately two hours, overrunning the planned schedule due to the continuous interest of the participants.

A follow-up panel discussion on CPS architectures will take place in the forthcoming IEEE Conference on Decision and Control, on December 13, in Florence, Italy. It has been organized and will be moderated by Professor Baras with the following invited panelists: Dr. Manfred Broy (Technical University Munich, Germany), Dr. David Corman (National Science Foundation (NSF), CPS Program Director, USA), Dr. Karl Henrik Johannson (Royal Institute of Technology (KTH), Sweden), Dr. P.R. Kumar (Texas A&M University, USA), Dr. Max Lemke (European

Commission, Complex Systems & Advanced Computing Head), Dr. Alberto Sangiovanni-Vincentelli (University of California Berkeley, USA), Dr. Vijay Srinivasan (National Institute of Standards and Technology (NIST), USA).

To achieve superior levels of performance, CPS architectures will need to be highly integrated, be able to easily adapt to rapidly changing requirements and environmental conditions, and CPS systems will need to be agile. The use of integrated system architectures changes the very nature of MBSE because loosely coupled design flows are replaced by chains of many-to-many relationships between the system stakeholders, their design concerns, viewpoints, views and models. Stringent requirements on system agility imply that complex systems will have connectivity relationships that allow for systematic assembly (or composition) from simpler systems. Design space exploration and trade studies are more difficult to conduct because:

(1) System relationships can reach laterally across systems hierarchies and/or intertwined network structures; and

(2) Ideal architectural solutions to integration and agility conflict.

System validation is more difficult because system components will be required to serve multiple functions, and cause-and-effect mechanisms are no longer localized and obvious. The tenet of our approach is that these CPS design challenges can be met through the use of design flows and operational processes that are strategic in their use of top-down hierarchical decomposition (to simplify the description and solution of problems), bottom-up composition (to allow for increased system agility and reliability, and decreased time-to-deployment), abstraction (to remove problem details not immediately relevant to decision making) and formal methods (to ensure that models of system functionality, system design, and decision making are correct).

3. Model-Based Systems Engineering for CPS

Model-based Systems Engineering (MBSE) [8] has emerged as a promising methodology for the systematic design, performance evaluation and validation of complex engineering systems. "(MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [8]. MBSE is a relatively new development in the Systems Engineering technical community which emphasizes the practice of systems development through the use of models (of all types). MBSE facilitates the flow of requirements through models, a methodology that is at the same time compact and enforces consistency between data and requirements (through the models). Figure 1 describes the basic steps of the MBSE process that we have developed, and have been teaching at the University of Maryland (UMD) for several years [9]. This MBSE process has the following steps (phases): Requirements Collection, Construction of System Structure Model (what the system consists of), Construction of System Behavior Model (what the system does), Mapping of Behavior onto Structure (what structure components will perform parts of behavior), Allocation of Requirements to Structure and Behavior Components, Trade-Off Analysis, Validation and Verification.

As illustrated in Figure 1, the process moves between these steps in an iterative manner, until satisfactory alternative system designs are developed. The process is executed at different levels of granularity (detail/aggregation). As the MBSE process executes a *system architecture* is developed through the creation of behavior and structure components, their interrelationships and the allocation of behavior components to structure components. At this point if one wishes to utilize an Architecture Design Language and associated tool to capture the system architecture, she/he can do so, but it is not necessary.

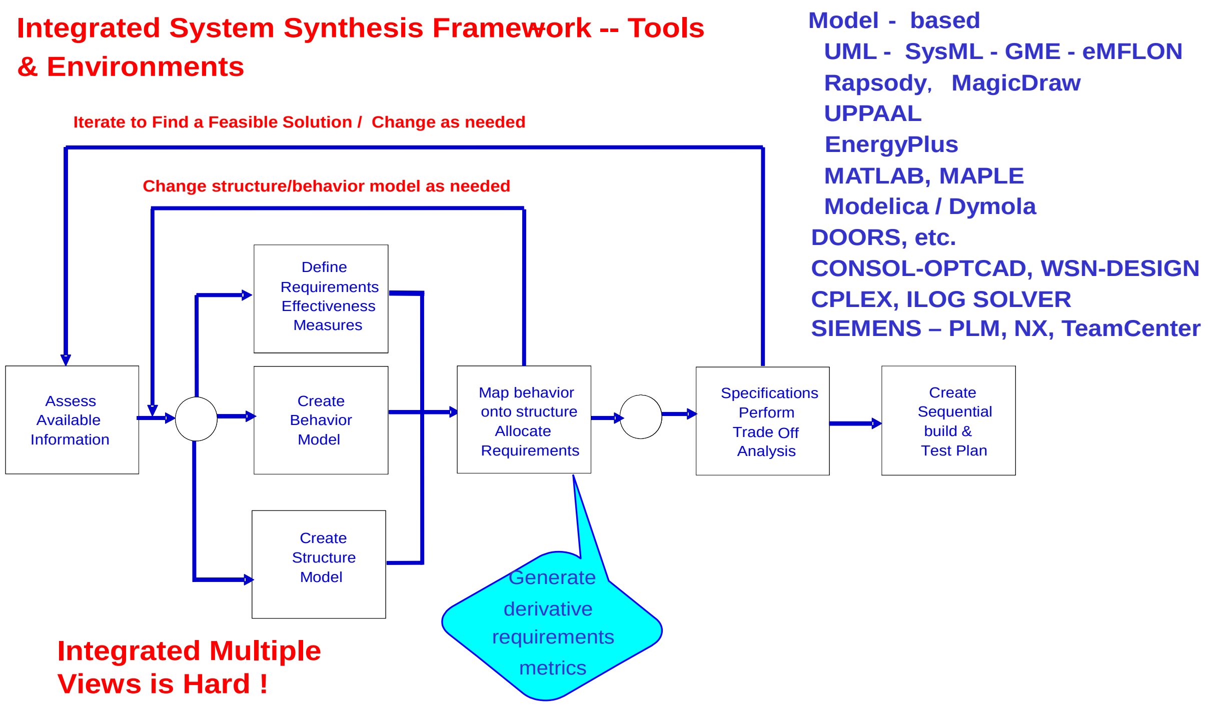

Integrated System Synthesis Framework -- Tools & Environments
Iterate to Find a Feasible Solution / Change as needed
Change structure/behavior model as needed
Model - based
UML - SysML - GME - eMFLON
Rapsody, MagicDraw
UPPAAL
EnergyPlus
MATLAB, MAPLE
Modelica / Dymola
DOORS, etc.
CONSOL-OPTCAD, WSN-DESIGN
CPLEX, ILOG SOLVER
SIEMENS – PLM, NX, TeamCenter
Assess Available Information
Define Requirements Effectiveness Measures
Create Behavior Model
Create Structure Model
Map behavior onto structure Allocate Requirements
Specifications Perform Trade Off Analysis
Create Sequential build & Test Plan
Generate derivative requirements metrics
Integrated Multiple Views is Hard !

Fig. 1. Model-Based Systems Engineering Process [9]

High levels of MBSE productivity will be achieved through the use of high-level visual abstractions coupled with lower-level (mathematical) abstractions suitable for formal systems analysis. Recent research has demonstrated the use of SysML as a centerpiece abstraction for team-based system development, with a variety of interfaces and relationship types (e.g., parametric, logical and dependency) providing linkages to detailed discipline-specific analyses and orchestration of system engineering activities.

3.1 Systems Modeling Language (SysML)

SysML [10] is a general purpose graphical modeling language that was developed based on UML and is a key enabler for the MBSE process by providing ways for the representation and analysis of complex engineering systems. SysML supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities. SysML supports model and data interchange via XML Metadata Interchange (XMI) and the AP233 standard. Recent research has demonstrated the use of SysML [10] as a centerpiece abstraction for team-based system development, with a variety of interfaces and relationship types (e.g., parametric, logical and dependency) providing linkages to detailed discipline-specific analyses and orchestration of system engineering activities. The four fundamental pillars of SysML are the support of models for the *structure* of the system, models of the *behavior* of the system, models for capturing the requirements for the system via the new system, which ties design variables and metric parametric representations to the structure and

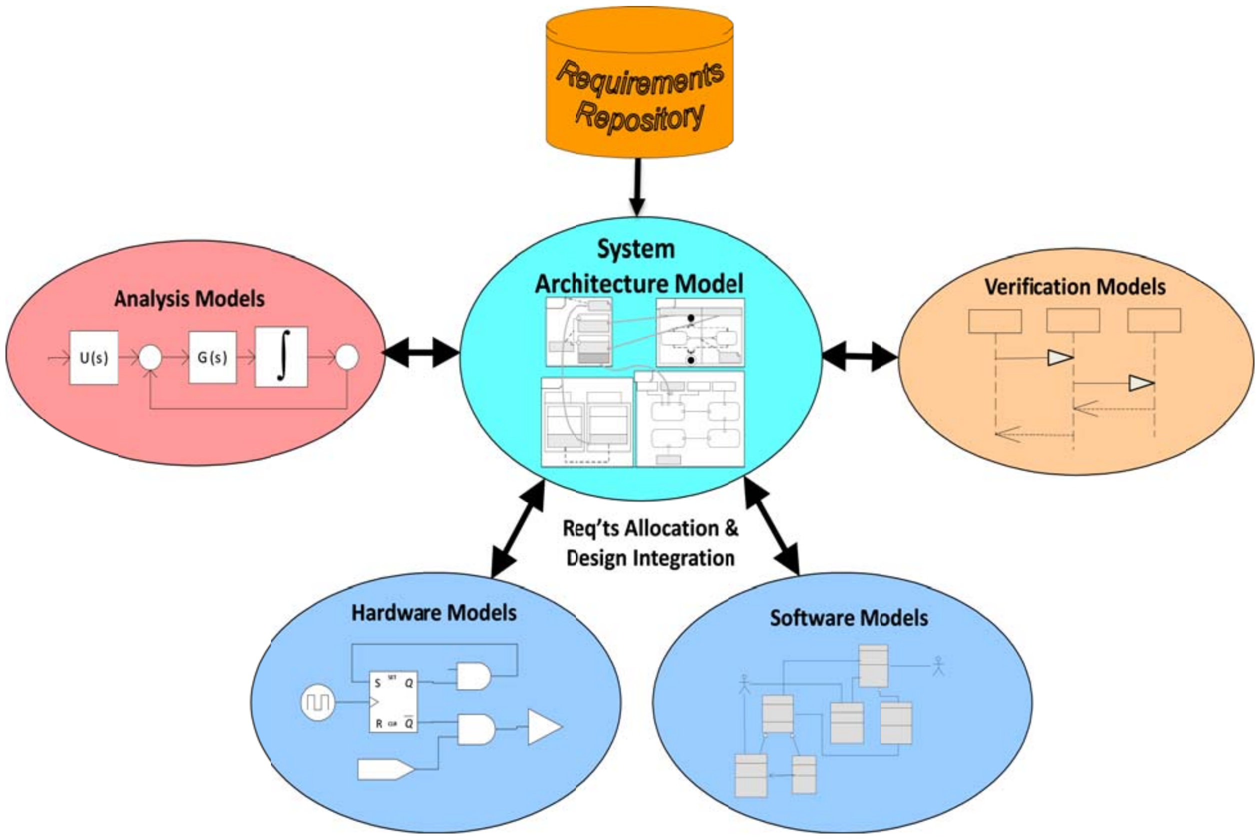


Fig. 2. Multi-domain model integration via system architecture model (SysML)

behavior models (a kind of annotation of these models). Parametric diagrams are the key to linking SysML-based system models to analysis models, including trade-off analysis models such as multimetric optimization (e.g. IBM-ILOG CPLEX) and constraint based reasoning tools (e.g. IBM-ILOG Solver). SysML, as a language for describing the system architecture, is a catalyst for the integration of various modeling environments, as well as analysis/design environments, for complex systems, while allowing multiple disciplinary views of the system and its components (Figure 2), where the System Architecture Model is described via SysML.

Our research in this task has established a SysML-based approach towards the development of the foundations for the ***model integration framework for CPS***. A key component in the proposed Framework are the resulting **CPS modeling integration hubs** (Figure 4).

3.2 CPS Modeling Integration Hub Architecture

A major challenge in MBSE for CPS is to have models that are consistent with each other. However, besides having consistent data there is a need for the models to work together in order to offer a holistic Systems Engineering approach to the designer of CPS. SysML is used in the core of our modeling integration hub (Fig. 2 and Fig. 4). The main aim is to integrate this core module with external tools, each one used in a different phase of the Systems Engineering process [9, 10, 11].

Another key component of the emerging Framework is a ***metamodeling environment*** with its associated languages and semantics based on sophisticated versions of annotated block diagrams and bond graphs [11]. A metamodeling layer (Figure 3) stands one abstraction layer above the actual design implementation in a modeling language. A metamodel consists of the constructs of a modeling language together with the rules that specify the allowable relationships between these constructs. It can be considered as the grammar of that modeling language. At the metamodeling layer ***model transformations*** take place (Figure 3). There are many alternatives in terms of model transformation tools, like ATL, GME, eMoflon, QVT. In our research the eMoflon model transformation tool was used [12], [13].

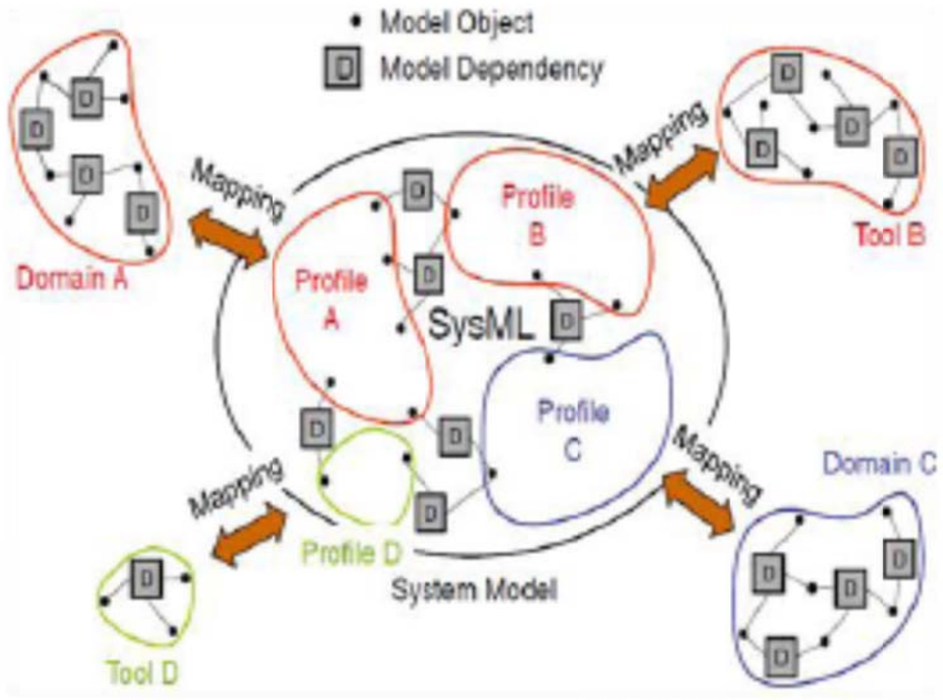


Fig. 3: System modeling transformation

The resulting MBSE ***system modeling environment*** can be thought of as a “virtual” product line management (PLM) environment for CPS, across discipline tools. To achieve this integration a three layer approach needs to be followed. Initially, for the tool we need to integrate, a domain specific profile is created in SysML [14]. Then a model transformation is defined, followed by the implementation of tool adapters that are used as a middleware for exchanging information between the model transformation layer and the other components of the hub. Tool adapters work as the “glue” between the different pieces of software. Their role is to access/change information inside a model and call the appropriate Java functions generated by the eMoflon tool to perform model transformations [11], [14]. Fig. 4 presents these layers as well as the areas for which we need to integrate tools with the core module to realize the MBSE vision of a system design experience for CPS.

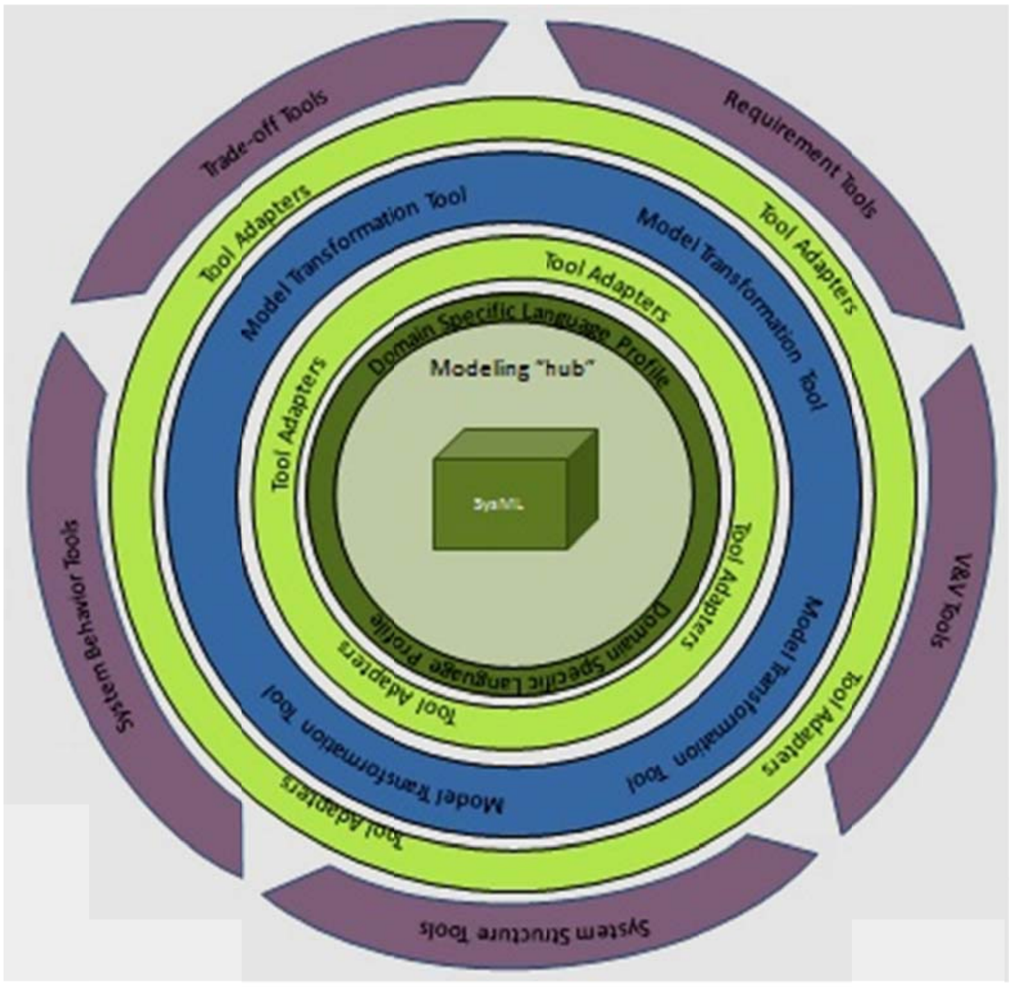


Fig. 4. The CPS modelling hub

In our research to date we have successfully integrated (and demonstrated the use of, in real industrial CPS problems) various environments with SysML: Modelica, MATLAB (Stateflow / Simulink, Mathematica, Maple, COMSOL etc.). CPS puts additional significant challenges in this integration of models and views due to the fundamental heterogeneity of CPS components and the hybrid (logic-analog nature of CPS). Our research to date has also addressed the development of new mathematical foundations for this model integration and towards a framework for standardization in these so-called ***CPS modeling integration hubs***. So far we have developed such CPS modeling integration hubs for power grids, microrobotics, energy efficient buildings, wireless sensor networks, wireless network protocols and vehicle management systems for next generation all-electric aircraft.

4. Tradeoff Analysis and Design Space Exploration

Although progress to date in MBSE facilitates the integration of system component models from different domains, we still need an integrated environment to optimize system architecture, manage the analysis and optimization of diverse measures of effectiveness (MoE), manage the various acceptable designs and most than anything else perform tradeoff analysis. Tradeoff is an essential part of system design, as it implements design space exploration. SysML does not provide a way for engineers to formally evaluate and rank design criteria, conduct sensitivity analysis, search design spaces for better design solutions, and conduct trade studies. To address this challenge we have introduced [3, 7, 11] the concept that SysML needs to be integrated with industrial-strength multi-objective algorithms, constraint-based reasoning algorithms, with appropriate linkages to modeling/simulation environments (see Figure 5). An integration of SysML with a tradeoff tool will allow the designer to make decisions faster and with more confidence.

4.1 Integration of SysML-Integrated CPS Modeling Hubs with Tradeoff Tools

We have recently developed and demonstrated [11] the first ever integration of a powerful tradeoff analysis tool (and methodology), Consol-Optcad, which is a sophisticated multi-criteria optimization tool developed at the University of Maryland, with our SysML-based modeling integration hubs for CPS. Consol-Optcad is a multi-objective optimization tool that allows interaction between the model and the user. It can handle non-linear objective functions and constraints with continuous values. Another version of Consol-Optcad has been developed to handle also logical variables, via integer and constraint programming [15].

In systems development and after the system structure is defined there is a need to calculate the design parameters that best meet the objectives and constraints. Usually when we deal with complex systems and optimization is under consideration, this is not a trivial task. The support of an interactive tool, like Consol-Optcad, to help the designer resolve the emerging trade-offs is necessary. A major advantage of Consol-Optcad is that it allows the user to interact with the tool, while the optimization is under way. The designer might not know or might not be in a position at the beginning to specify what preferred design means. Therefore such interaction with the tool could be of great benefit [15], [16]. Another key feature of Consol-Optcad is the use of the Feasible Sequential Quadratic Programming (FSQP) algorithm for the solver [16, 17]. FSQP's advantage is that as soon as we get an iteration solution that is inside the feasible region, feasibility is guaranteed for the following iterations as well. Moreover, very interesting is the fact that besides traditional objectives and constraints Consol-Optcad allows the definition of functional constraints and objectives that depend on a free parameter. Consol-Optcad has been applied to the design of flight control systems [17], rotorcraft systems [18, 19], integrated product process design (IPPD) systems [15] and other complex engineering systems.

For effective design space exploration and tradeoff analysis it is important to have the ability o compute sensitivities to proposed changes and evaluate "what if" types of questions. CONSOL-OPTCAD is such a sophisticated multi-criteria optimization tool, which incorporates duality methods of analysis (involving both numerical and discrete variables) for problems such as IPPD, as well as innovative visualization techniques to help engineers understand the impact of design choices (see for instance the *Pcomb* diagram in Figure 6(a) from [11]). This is unique feature of Consol-Optcad as compared to other approaches currently like the one use in Model-Center. Figure 6(b) shows the situation with a functional requirement; another unique strength of Comsol-Optcad.

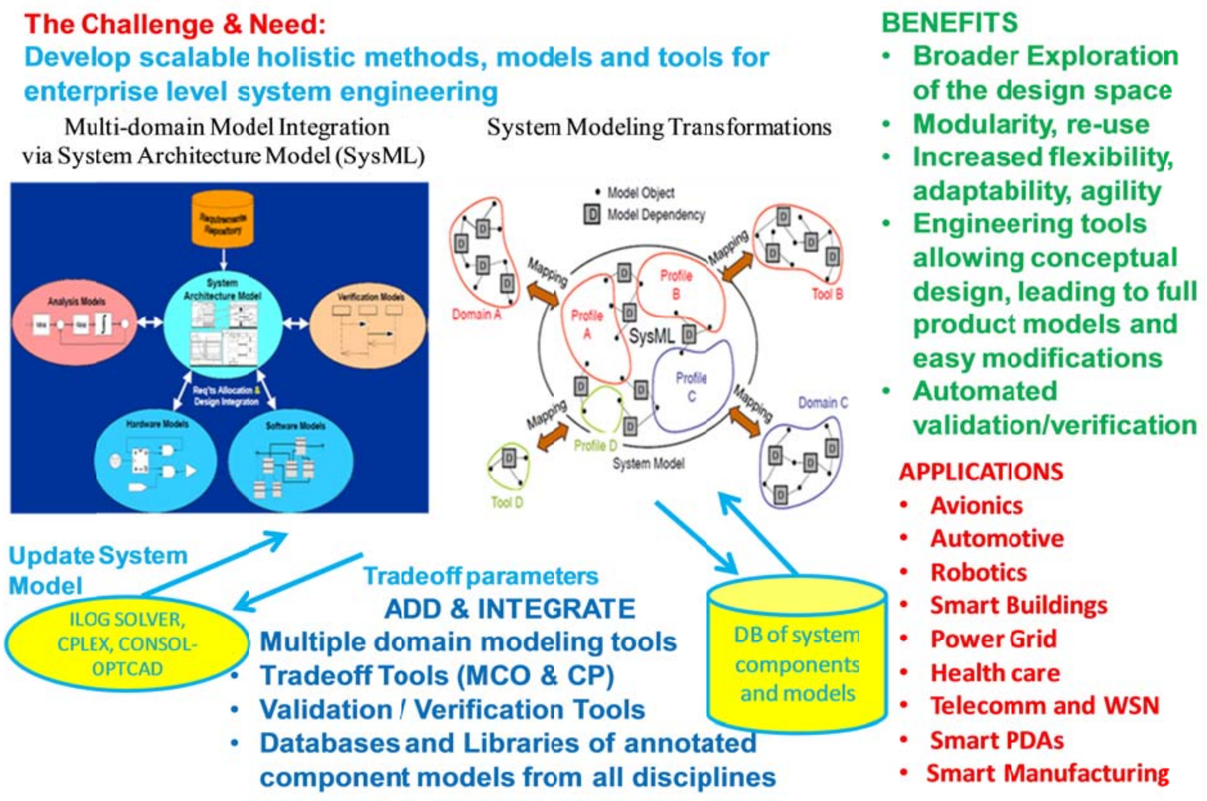
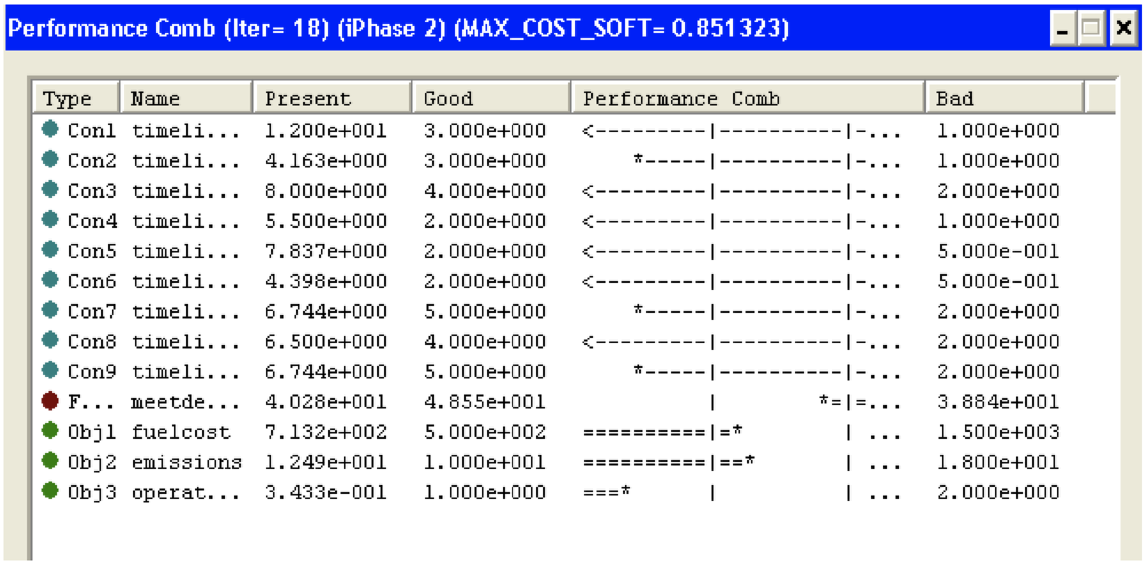

The Challenge & Need:
Develop scalable holistic methods, models and tools for enterprise level system engineering
Multi-domain Model Integration via System Architecture Model (SysML)
System Modeling Transformations
Requirements Repository
System Architecture Model
Analysis Models
Verification Models
Req'ts Allocation & Design Integration
Hardware Models
Software Models
Model Object
Model Dependency
Domain A
Mapping
Profile A
Profile B
SysML
Profile C
Profile D
System Model
Tool B
Tool D
Domain C
BENEFITS
Broader Exploration of the design space
Modularity, re-use
Increased flexibility, adaptability, agility
Engineering tools allowing conceptual design, leading to full product models and easy modifications
Automated validation/verification
APPLICATIONS
Avionics
Automotive
Robotics
Smart Buildings
Power Grid
Health care
Telecomm and WSN
Smart PDAs
Smart Manufacturing
Update System Model
ILOG SOLVER, CPLEX, CONSOL-OPTCAD
Tradeoff parameters
ADD & INTEGRATE
Multiple domain modeling tools
Tradeoff Tools (MCO & CP)
Validation / Verification Tools
Databases and Libraries of annotated component models from all disciplines
DB of system components and models

Fig. 5: Linking Design Space Exploration Tools with SysML – Integrated CPS Modeling Hub


Performance Comb (Iter= 18) (iPhase 2) (MAX_COST_SOFT= 0.851323)

```
Type   Name        Present      Good         Performance Comb              Bad
● Con1 timeli...   1.200e+001   3.000e+000   <---------|----------|-...    1.000e+000
● Con2 timeli...   4.163e+000   3.000e+000       *-----|----------|-...    1.000e+000
● Con3 timeli...   8.000e+000   4.000e+000   <---------|----------|-...    2.000e+000
● Con4 timeli...   5.500e+000   2.000e+000   <---------|----------|-...    1.000e+000
● Con5 timeli...   7.837e+000   2.000e+000   <---------|----------|-...    5.000e-001
● Con6 timeli...   4.398e+000   2.000e+000   <---------|----------|-...    5.000e-001
● Con7 timeli...   6.744e+000   5.000e+000       *-----|----------|-...    2.000e+000
● Con8 timeli...   6.500e+000   4.000e+000   <---------|----------|-...    2.000e+000
● Con9 timeli...   6.744e+000   5.000e+000       *-----|----------|-...    2.000e+000
● F... meetde...   4.028e+001   4.855e+001             |         *=|=...   3.884e+001
● Obj1 fuelcost    7.132e+002   5.000e+002   ==========|=*        | ...    1.500e+003
● Obj2 emissions   1.249e+001   1.000e+001   ==========|==*       | ...    1.800e+001
● Obj3 operat...   3.433e-001   1.000e+000   ===*      |          | ...    2.000e+000
```

Fig. 6(a). Pcomb after the 18th iteration

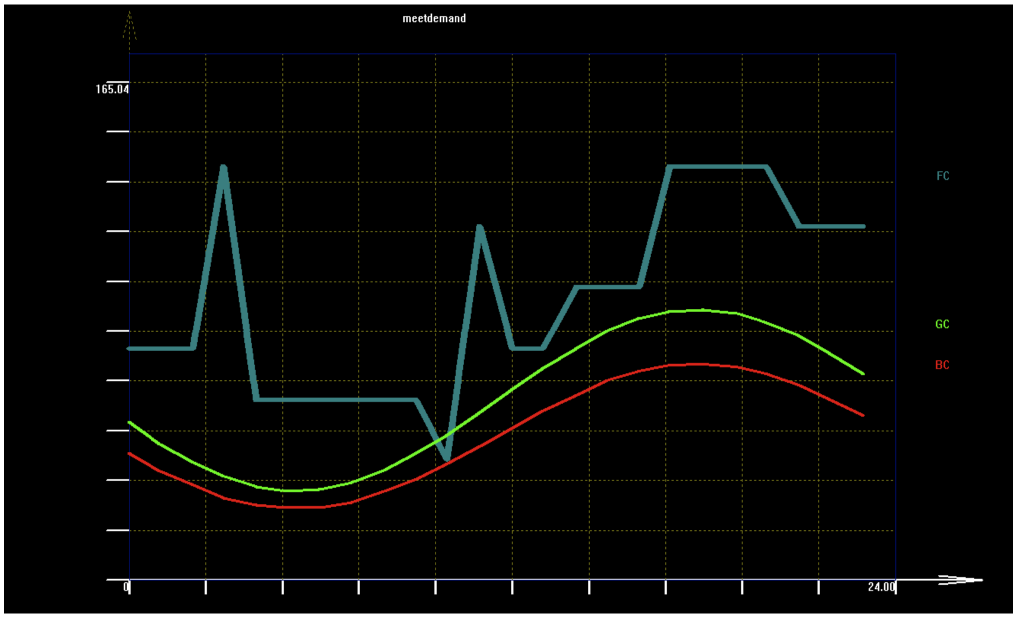


Fig. 6(b). Functional constraint after 18th iteration

The details of the integration framework, and the separate steps that were followed to achieve the integration between SysML (from MagicDraw [14]) and Consol-Optcad, can be found in [11]. SysML is not a tool specific language, but MagicDraw was used because it is more open than other tools and it can be modified more easily.

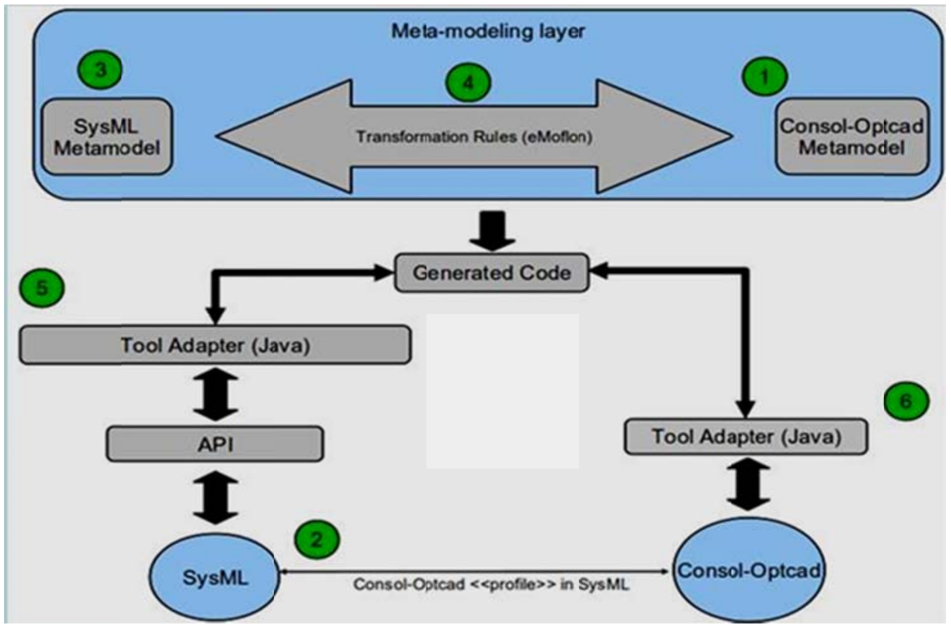


Fig. 7. Integration Framework

Fig. 7 presents the architecture of the integration together with numbered steps that need to be followed to complete the integration process. According to the three-layer approach described earlier, the integration process is divided into three main parts. The first part concerns the mapping of the objects between the two languages (SysML, Consol-Optcad). It also includes the development of specific semantics that are used for that purpose; in this case a profile of Consol-Optcad in SysML was created. The second part is the meta-modeling layer where the transformation between the two models takes place. The last part consists of implementing the appropriate tool adapters. The creation of a Consol-Optcad profile is the first step of the integration process. Profiling is the mechanism that SysML has to allow the designer to use additional constructs inside the development environment. After a profile is being built and since it gives the user the ability to use constructs of a specific tool directly in SysML, it decreases significantly the design effort. A SysML Profile is composed by a set of stereotypes and their relationships [14]. Each Consol-Optcad construct is represented in the profile diagram by a stereotype, according to the Consol-Optcad specification document [15, 16, 17]. After the profile has been created the designer can load the new profile in the project and start using it by simply dragging and dropping Consol-Optcad constructs in the block definition diagram area. This integration process is also shown in Figure 8. We would like to emphasize that the steps described are generic, in the sense that they can be easily generated for different instantiations of both the CPS model integrated hub and the tradeoff and design exploration tool.

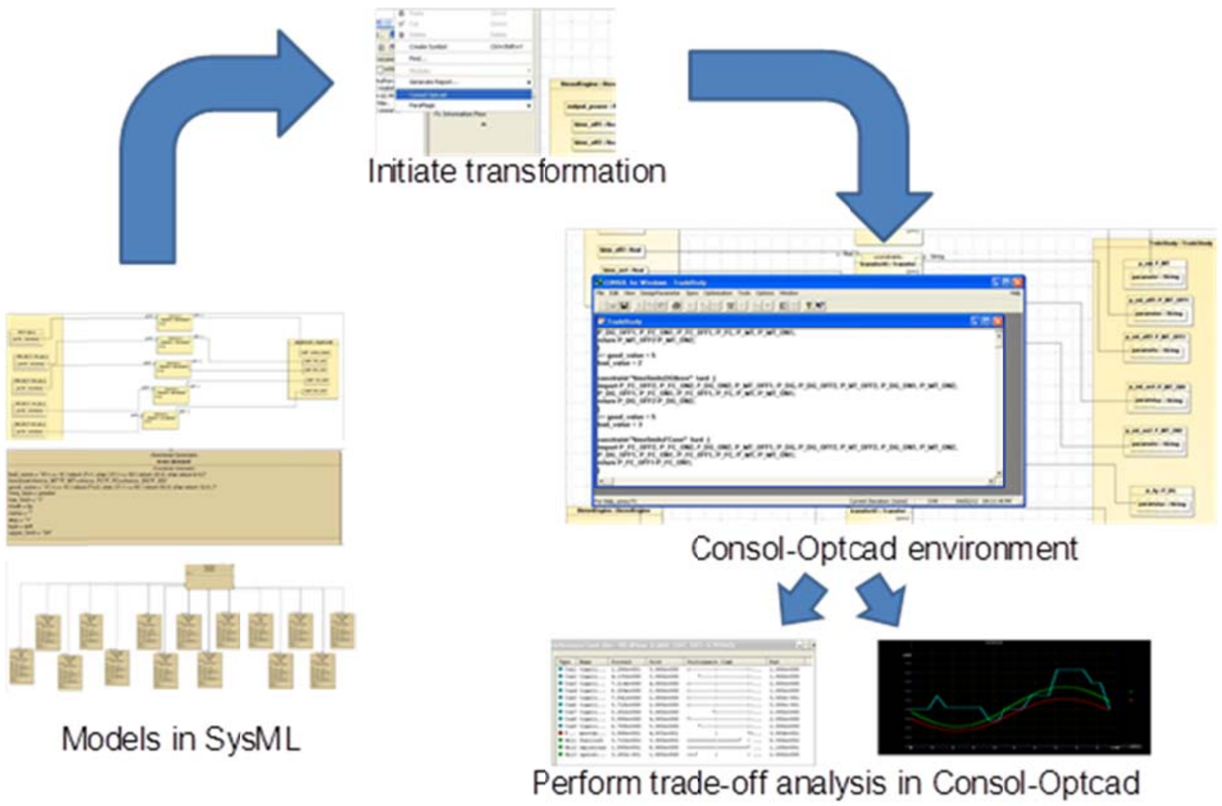

Initiate transformation
Consol-Optcad environment
Models in SysML
Perform trade-off analysis in Consol-Optcad

Fig. 8: Illustrating the steps used in the integration of the CPS IMH with the tradeoff tool.

The linkage of tradeoff analysis and design space exploration tools in the proposed Framework should be made through connecting to the Requirements Diagram (RD) and the Parametric Diagram (PD); both new diagrams introduced by SysML. This is illustrated in Figure 9. This is an important concept in the proposed Framework

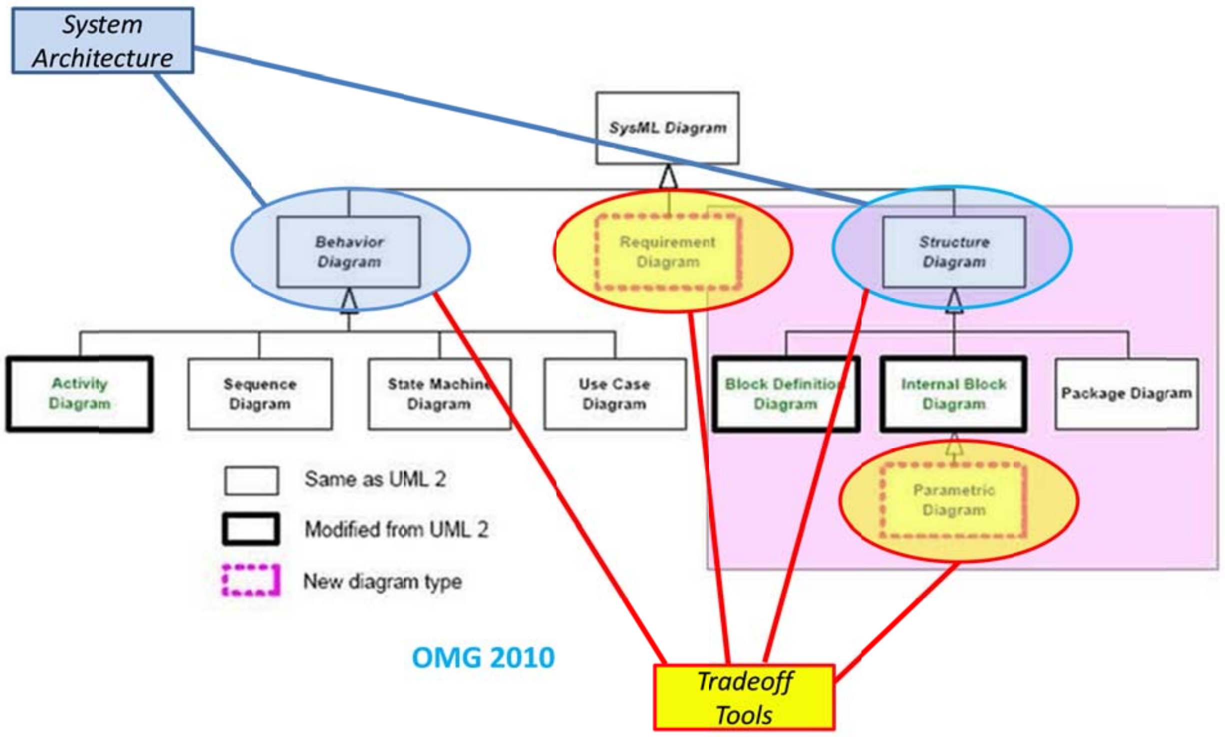


Fig. 9: Integrating Tradeoff and Design Space Exploration tools with SysML

4.2 Integration of Constrained-Based Reasoning and Optimization for CPS Tradeoff Analysis and Design Space Exploration

Requirements in a complex system, including CPS, will eventually be represented as constraints and metrics with logical and numerical variables and values. These constraints and metrics can include also time. Thus an important component in the proposed Framework is that tradeoff methods and tools should be able to freely move form constraints to metrics and conversely, as indeed metrics and constraints are mathematical dual objects. This involves the integration of constraint based programming (CP) with multi-objective constrained optimization (mixed integer-numerical).

In constraint programming [20, 21, 26, 33] there are two problems: (1) modeling the problem(s) of a certain domain, and (2) finding a 'good' (high-performance) solution algorithm that solves those problems efficiently. The models are expressed using logic formulae over Booleans, linear or non-linear mathematical (in)equalities over integers or reals, or set theoretical expressions; often all together. A solution to a constraint problem is a valuation for the free variables in the formulae such that all formulae are satisfied. The issues with modeling are that: (1) there could be many different ways to model a problem, and (2) the choice of modeling approach determines the solver algorithm used. Thus, not only the most expressive modeling approach should be chosen, but it also has to fit with the solver algorithms used. Note that the modeling 'language' does not have to be the same as the language (data structures) used in the solvers. A modeling language can use much higher-level, domain-specific constructs, from which all the lower-level, solver-oriented formulae could be automatically generated.

For solvers one can use purely mathematical algorithms (e.g. linear [27] or non-linear programming [28]) or algorithms developed by the AI community (e.g. constraint propagation/distribution over finite domains [29]) or algorithms that work on the symbolic representations of Booleans (e.g. manipulations on ordered binary decision diagrams [30]). The available packages such a ILOG solver [22], CHIP [23], ECLiPSe [24] and Prolog IV [25] are all robust and result of years of research in this field and can be used as solvers. They all have library of methods for solving the problem, but they also allow users to develop their own algorithms.

A further refinement on the constraint programming paradigm is the introduction of soft constraints [32], where not all, but the majority of the formulae must be satisfied by a solution. In the classical constraint programming paradigm all formulae must be satisfied by the solution, otherwise there is no solution. Soft constraints also allow assigning priorities and preferences to formulae, thus preferring solutions that satistfy more important constraints than less important ones. This sort of prioritization is a very powerful and pragmatic modeling technique, better reflecting reality than the classical, hard constraints.

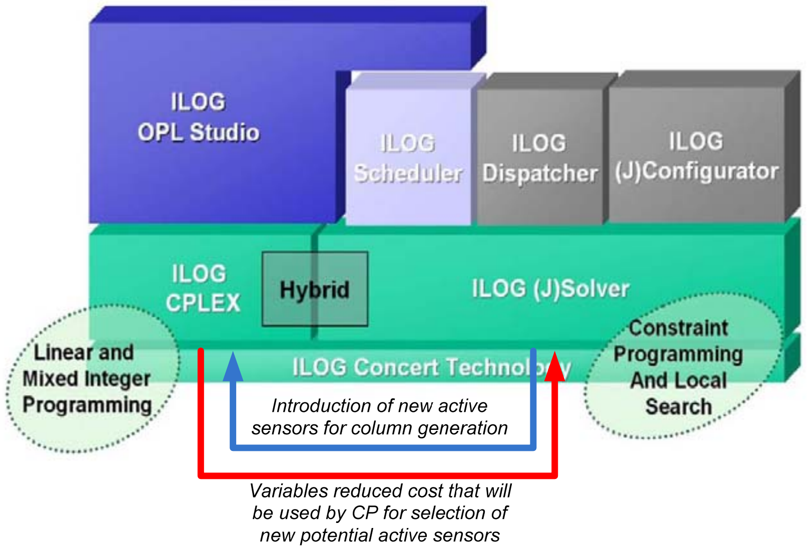


Fig. 10: the IBM-ILOG Optimization Suite

A good example of such an integration is the IBM-ILOG optimization suite. IBM-ILOG CPLEX and IBM-ILOG Solver form the core optimization engines for the platform. IBM-ILOG CPLEX provides powerful C and C++ fundamental algorithm libraries for operations research nonlinear programming professionals. These libraries include ILOG's simplex, barrier and mixed integer optimizers for linear, integer and quadratic programming. IBM-ILOG CPLEX also provides easy-to-use C++ modeling objects that allow the expression of linear and integer programs in a simplified form directly related to their algebraic models. The IBM-ILOG Solver is one of the core C++ libraries in the ILOG Optimization Suite and implements the basic engine for constraint-based optimization. It can solve highly combinatorial real-world problems that are impractical to solve with traditional mathematical programming methods. This high-performance constraint-programming engine can be used alone, or with the IBM-ILOG CPLEX. Using these engines one can also develop customized algorithms. In Figure

10, we have also shown the information that should be passed among the two engines to solve an example networked CPS problem, involving sensor networks.

One of the main goals of modular system design in general and structural software programming in particular is separation of concerns. In a component based design, separation of concerns leads to breaking the system into components that overlap in functionality as little as possible. Unfortunately, there are some concerns that cannot be localized and dealt with in a single component. These types of concerns are called cross-cutting concerns. Good examples of cross-cutting concerns for distributed systems are security issues, synchronization requirements, fault detection and intrusion detection. Aspect-oriented design methodology [34] is a systematic solution for coping with cross-cutting concerns. In component based architectures, one can represent aspects as separate components. In this way, while we are implementing functional components in a CPS, we do not need to explicitly address the aspect concerns. Instead, the system should offer implicit invocation mechanisms for invoking behavior in the functional component (such as routing) whose implementers were unaware of the concern (such as security). In this way, if the security requirements change we can go ahead and design and /or use a new security component (aspect). If the system is designed based on the aspect oriented design paradigm, it should be clear which one of the components can work under the new aspect requirements with minimal modifications.

The tradeoff analysis methodology that we include in the proposed Framework is based on the integrated and interoperable use of constrained based reasoning and multi-criteria optimization. It is capable of performing trade-off analysis for both the behavioral and the structural model of a system and its components, as well as of the allocation of behavioral components to structural components. One example instantiation is described in [11]. Design space exploration is based on effective tradeoff tools.

The integration and its implementation, as described in this report, was successfully applied to analyze a multi-criteria optimization problem concerning power allocation and scheduling in a microgrid [11]. Expanding the capabilities of this integration by making Consol-Optcad able to handle mix integer problems is currently under development, which represent the majority of problems that industry usually faces. Finding a way to incorporate structural changes and geometry to the design space exploration process is another very challenging task that can expand the usefulness of the integration presented (see section 6 below). Finally, another instantiation of the Framework is to integrate IBM CPLEX and IBM-ILOG Solver in our CPS integrated modeling hub -- tools that are used widely in industry with excellent results in many domains.

With the integration of design space exploration tools our proposed Framework addresses the fundamental CPS challenge of connecting multiple development environments, so as to provide a unified system view, while at the same time facilitating holistic (i.e. system level traceability and impact analysis). This accomplishes system architecture management across disciplinary domains. The Framework derived by our research, and proposed herein, represents a substantial and innovative extension of the current state of the art in Model-Based Engineering (MBE). Our approach and results to date address the following applications and challenges for CPS synthesis: (a) Broader exploration of the design space; (b) Dramatically increased flexibility and adaptability to changing environments, without time-consuming redesign; (c) Need for modifiable systems, reconfigurable or upgradable by reference to virtual models, by plug-replacing subcomponents; (d) Heterogeneous CPS model integration; (e) Engineering tools, technologies and methods that enable conceptual design – system design and production, that are useful for full product models and allow easy modification and upgrades.

5. Functional Mock-up Interface (FMI)

In the last two years the ***Functional Mock-up Interface*** (or ***FMI***) framework [35], for co simulation of complex systems has been gaining acceptance. Functional Mock-up Interface (FMI) is a tool independent standard to support both model exchange and co-simulation of dynamic models using a combination of XML-files and compiled C-code. The first version, FMI 1.0 (downloads#version1), was published in 2010. The FMI development was initiated by Daimler AG with the goal to improve the exchange of simulation models between suppliers and OEMs. As of today, development of the standard continues through the participation of 16 companies and research institutes (development). FMI is supported by over 35 tools (tools) and is used by automotive and non-automotive organizations throughout Europe, Asia and North America. The FMI specifications (http://www.modelisar.com/fmi.html) are distributed under open source Licenses. Each FMU (functional mock-up unit) model is distributed in a zip file with the extension ".fmu" which contains:

(i) An XML file containing among other things the definition of the variables used by the FMU;

(ii) All the equations used by the model (defined as a set of C functions);

(iii) Optional other data, such as parameter tables, user interface, documentation which may be needed by the model.

FMI defines a standardized interface to be used in computer simulations to develop complex CPS [35, 36, 37]. The vision of FMI is to support this approach: if the real product is to be assembled from a wide range of parts interacting in complex ways, each controlled by a complex set of physical laws, then it should be possible to create a ***virtual product*** that can be assembled from a set of models that each represent a combination of parts, each a model of the physical laws as well as a model of the control systems (using electronics, hydraulics, digital software, ..) assembled digitally. The FMI standard thus provides the means for model based development of systems and is used for example for designing functions that are driven by electronic devices inside vehicles (e.g. ESP controllers, active safety systems, combustion controllers). Activities from systems modelling, simulation, validation and test can be covered with the FMI based approach.

The four required FMI aspects of creating models capable of being assembled have been covered in the Modelisar project:

- FMI for model exchange,
- FMI for co-simulation,
- FMI for applications,
- FMI for PLM (integration of models and related data in product life-cycle management).

In practice, the FMI implementation by a software modelling tool enables the creation of a simulation model that can be interconnected or the creation of a software library of component models called ***FMUs*** (***Functional Mockup Units***).

The typical FMI approach is described by the following stages:

- A modelling environment describes a product sub-system by differential, algebraic and discrete equations with time, state and step-events. These models can be large for usage in off-line or online simulation or can be used in embedded control systems;
- As an alternative, an engineering tool defines the controller code for controlling a vehicle system;
- Such tools generate and export the component in an FMU (Functional Mock-up Unit);
- An FMU can then be imported in another environment to be executed;
- Several FMUs can – by this way – cooperate at runtime through a co-simulation environment, thanks to the FMI definitions of their interfaces.

The FMI specifications (http://www.modelisar.com/fmi.html) are distributed under open source Licenses. Each FMU (functional mock-up unit) model is distributed in a zip file with the extension ".fmu" which contains:

(i) An XML file containing among other things the definition of the variables used by the FMU;
(ii) All the equations used by the model (defined as a set of C functions);
(iii) Optional other data, such as parameter tables, user interface, documentation which may be needed by the model.

FMI models have several advantages over Simulink S-Functions:

- S-Functions format is proprietary, whereas the FMI schema is licensed under a BSD license.
- The building blocks of S-Functions are much more complex than FMI, making it very difficult to integrate in simulators other than Simulink itself.
- Furthermore, the S-Functions format is specific to Simulink.
- S-Functions are not suited for embedded systems, due to the memory overhead of S-Functions.

We have included FMI in the proposed Framework due to the benefits described above. However, we emphasize that the FMI framework by itself helps only for simulating complex CPS systems and not in performing the entire MBSE process as described here. However integrated with the rest of the components of our framework it does provide some very useful functionalities. We have used FMI model integration within our framework already in several applications.

6. Multi-Physics Models

One of the major challenges in modeling CPS and for performing MBSE of CPS, is the heterogeneity of physics involved in CPS (see Figure 11). This is dramatically different form VLSI design for example, as the heterogeneity of physics require representation of different design logics (the rule implied by each physics involved in the CPS).

Modeling implies the activity of forming a mathematical representation, and its algorithmic and computational implementation of the system behavior regarding the relationship between input (stimulus) and output (response) in terms of state variables. Similarly, the term simulation implies the activity of actually utilizing the model produced by the modeling activity. That is to say, simulation implies the activity of predicting the behavior of the system according to the assignment of values to a subset of the associated state variables. The term “multiphysics” has been used historically in more than one undeclared contexts. Primary among these are: “Multifield” to denote the simultaneous excitation and response of the system by multiple physical fields; “multidomain” to denote the interaction among continuum representations of systems with drastically different properties (e.g. fluid-structure interaction, moving solidification boundary problems etc.) through sharable boundaries; “multi-scale” to denote the consistent bridging of various behavioral models of the system at hand, at various length scales as required by a multitude of scopes ranging from manufacturing process perspective to macro-behavioral utilization. Furthermore, it is significant to note that any combination of these three semantic possibilities generates four more meanings of the term multiphysics. Accordingly, this permits the construction of a conceptual attribute space that is spanned by the basis attributes, “multifield”, “multidomain”, and “multiscale”.

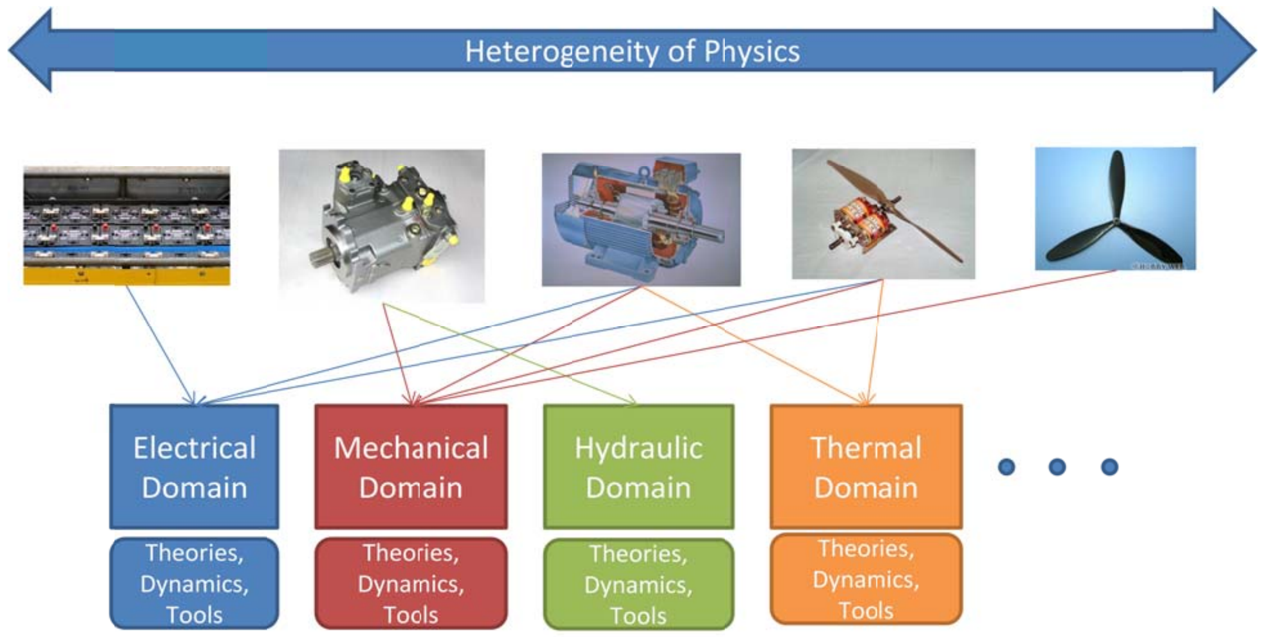

Heterogeneity of Physics
Electrical Domain
Theories, Dynamics, Tools
Mechanical Domain
Theories, Dynamics, Tools
Hydraulic Domain
Theories, Dynamics, Tools
Thermal Domain
Theories, Dynamics, Tools
● ● ●

Fig. 11: Illustrating multiple physical components and interactions in CPS

Semantically, a multiphysics system consists of more than one component governed by its own principle(s) for evolution or equilibrium, typically conservation or constitutive laws. A major classification in such systems is whether the coupling occurs in the bulk (e.g., through source terms or constitutive relations that are active in the overlapping domains of the individual components) or whether it occurs over an idealized interface that is lower dimensional or a narrow buffer zone (e.g., through boundary conditions that transmit fluxes, pressures, or displacements). Typical examples of bulk-coupled multiphysics systems with their own extensively developed literature include radiation with hydrodynamics in astrophysics (radiation-hydrodynamics, or “rad-hydro”), electricity and magnetism with hydrodynamics in plasma physics (magnetohydrodynamics), and chemical reaction with transport in combustion or subsurface flows (reactive transport). Typical examples of interface-coupled multiphysics systems are ocean-atmosphere dynamics in geophysics, fluid-structure dynamics in aeroelasticity, and core-edge coupling in tokamaks. Beyond these classic multiphysics systems are many others that share important structural features.

Success in simulating forward models leads to ambitions for inverse problems, sensitivity analysis, uncertainty quantification, model-constrained optimization, and reduced-order modeling, which tend to require many forward simulations. In these advances, the physical model is augmented by variables other than the primitive quantities in which the govern equations are defined. These variables may be probability density functions, sensitivity gradients, Lagrange multipliers, or coefficients of system-adaptive bases. Equations that govern the evolution of these auxiliary dependent variables are often derived and solved together with some of the physical variables. When the visualization is done in situ with the simulation, additional derived quantities may be carried along. Error estimation fields in adaptive meshing applications may constitute yet more. Though the auxiliary variables may not be "physical" in the standard sense, they give the overall simulation the structure of multiphysics. Still other systems may have a multiphysics character by virtue of being multirate or multiresolution. A chemical kinetics model may treat some components as being in equilibrium, idealizing a fast relaxation down to a constraint manifold on which other components vary more slowly. Some phenomena may be partitioned mathematically by projections onto wavelet bases of different frequency or wavenumber properties that are naturally treated differently. The current state of affairs in modeling and simulation of multi-physics systems cans be found in [38, 39].

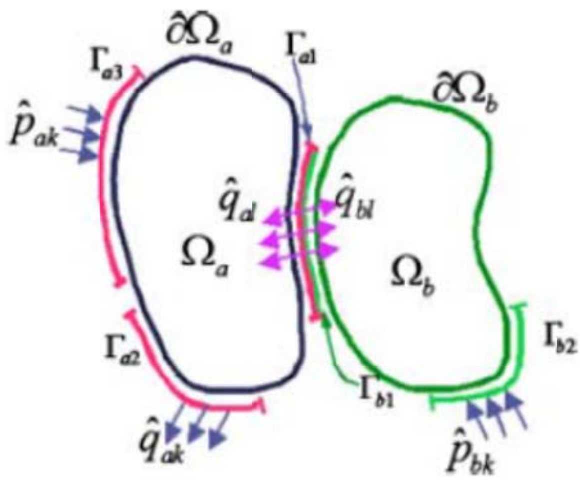


Fig. 12: Interacting multi-domain and multi-domain continuum systems across geometries

A key challenge in multi-physics modeling and simulation is the handling of geometry, constraints across geometry, geometry across scales (see Figure 12). Regarding CPS the interface of the components of multi physics models with the cyber models is also challenging. For these reasons we include in the proposed Framework languages such as Modelica, Dymola, COMSOL, associated tools and environments.

The Modelica (public domain) and Dymola (Dassaux Systemes) languages [40, 41]and associated tools, provide interesting capabilities in Systems Engineering and solutions for modeling and simulation, as it is possible to simulate the dynamic behavior and complex interactions between systems of many engineering fields, such as mechanical, electrical, thermodynamic, hydraulic, pneumatic, thermal and control systems. Thus users of Modelica/Dymola can build integrated models and have simulations results that depict reality. Both languages are very well suited for multi-physics continuum systems, but have weaknesses regarding the modeling of cyber and hybrid dynamical systems. However they do offer capabilities for modeling geometry, matter and associated deterministic and stochastic tolerances. In addition both model very well differential-algebraic equations that frequently appear in multi-physics models.

The COMSOL Multiphysics simulation environment [42, 43] facilitates all the steps in the modeling process – defining component or system geometry, meshing, specifying its physics, solving, and then visualizing the results. It also serves as a platform for application specific modules. Model set-up is quick, thanks to a number of predefined physics interfaces for applications ranging from fluid flow and heat transfer to structural mechanics and electrostatics. Material properties, source terms, and boundary conditions can all be spatially varying, time-dependent, or functions of the dependent variables. One can freely mix physics interfaces into new multiphysics combinations as well as couple with any application specific module. As an alternative to writing one’s own simulation code, the COMSOL Multiphysics user interface gives the option to specify one’s own partial or ordinary differential equations (PDEs or ODEs) and link them with other physics interfaces. When combined with the CAD Import Module or one of the LiveLink products, this enables one to run custom simulations on CAD models from industry-standard formats.

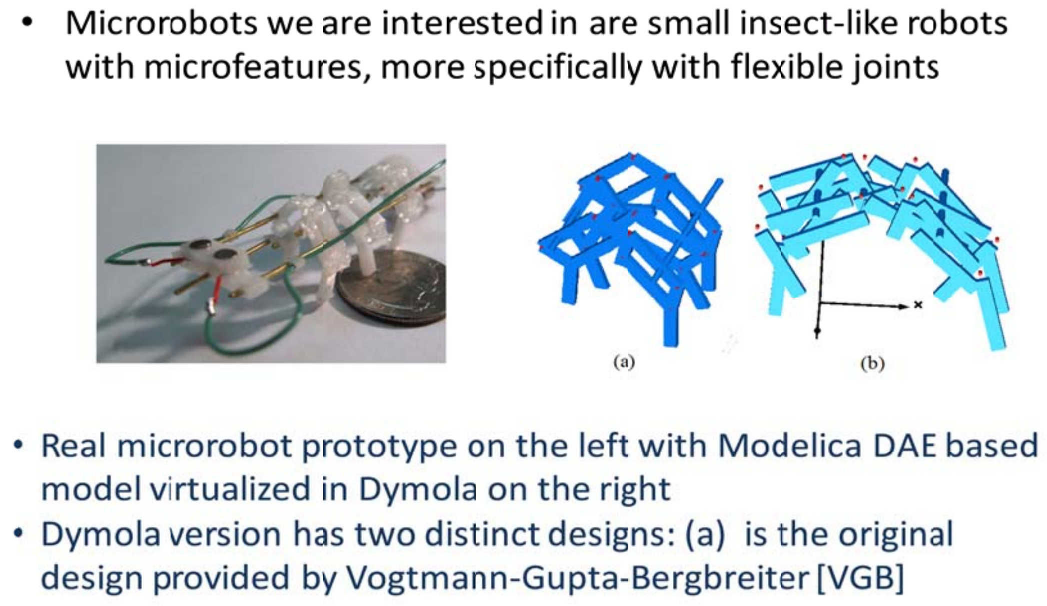
Both Modelica/Dymola and COMSOL integrate well with geometry modeling tools like CATIA (Dassaux Systems). They both integrate very well within the Framework described in Figures 2, 3, 4, 5. We have successfully used in CPS problems involving microgrids, microrobots and energy efficient buildings in our research so far.

7. Successful Applications of the Proposed Framework

In [11] we presented the CPS modeling hub as a way to realize the Model-Based Systems Engineering vision and face today's challenges on systems synthesis and development. Furthermore, we introduced a version of the proposed Framework for integrating the SysML-based CPS hub with Consol-Optcad. In [11] we provided details on how each step of the integration was implemented and what tools were used throughout this process. The SysML Consol-Optcad integration facilitates the problem formulation for the user and also enables the design and optimization processes, interacting and working in parallel in order to achieve the best possible design. A trade-off problem for an electrical microgrid was developed and solved to demonstrate the utility of the integration. Distributed Generation (DG) has emerged as a way to address shortcomings of power grids. In DG the generating systems are of small scale, their use is local and they are geographically distributed. However, DG can cause problems to the network, like reverse power flow, excessive voltage rise, increased fault levels, harmonic distortion and stability problems, due to their independent operation. To overcome such problems various distributed energy resources (DERs) are grouped together and together with loads to form what is called a *microgrid* [11]. The Energy Management System plays a central role in the smooth operation of microgrids; it makes the decisions about generation and distribution of electrical energy. These decisions are based on many factors, like power demand, weather, price of electricity and heat, fuel cost, emissions cost and government policies, to name a few. The DERs that take part in a microgrid can be electrical, thermal or a combination. Solar panels, small wind and hydro generators, micro turbines, diesel engines, fuel cells, gas turbines are some examples of DERs.

We defined a microgrid that consists of three power sources: one microturbine, one fuel cell and one diesel engine. The characteristics of each type of power source were realistic [11]. The microgrid was supposed to provide power to a residential building that has 50 apartments. We addressed the problem of finding an optimal solution in terms of scheduling and power output of each engine for a period of 24 hours. The optimal solution was sought while trying to minimize operational cost, fuel cost, emissions and meet customer demand. We assumed that each power source can be turned on and off only two times during a day, because of the costs associated with turning on/off power sources. Details can be found in [11].

In [44] we described this new methodology and environment for Cyber Physical Systems (CPS) synthesis and demonstrate it in the design of microrobots viewed as CPS. The microrobots of our study are shown in Figure 13. Various types of microrobots have been developed in recent years

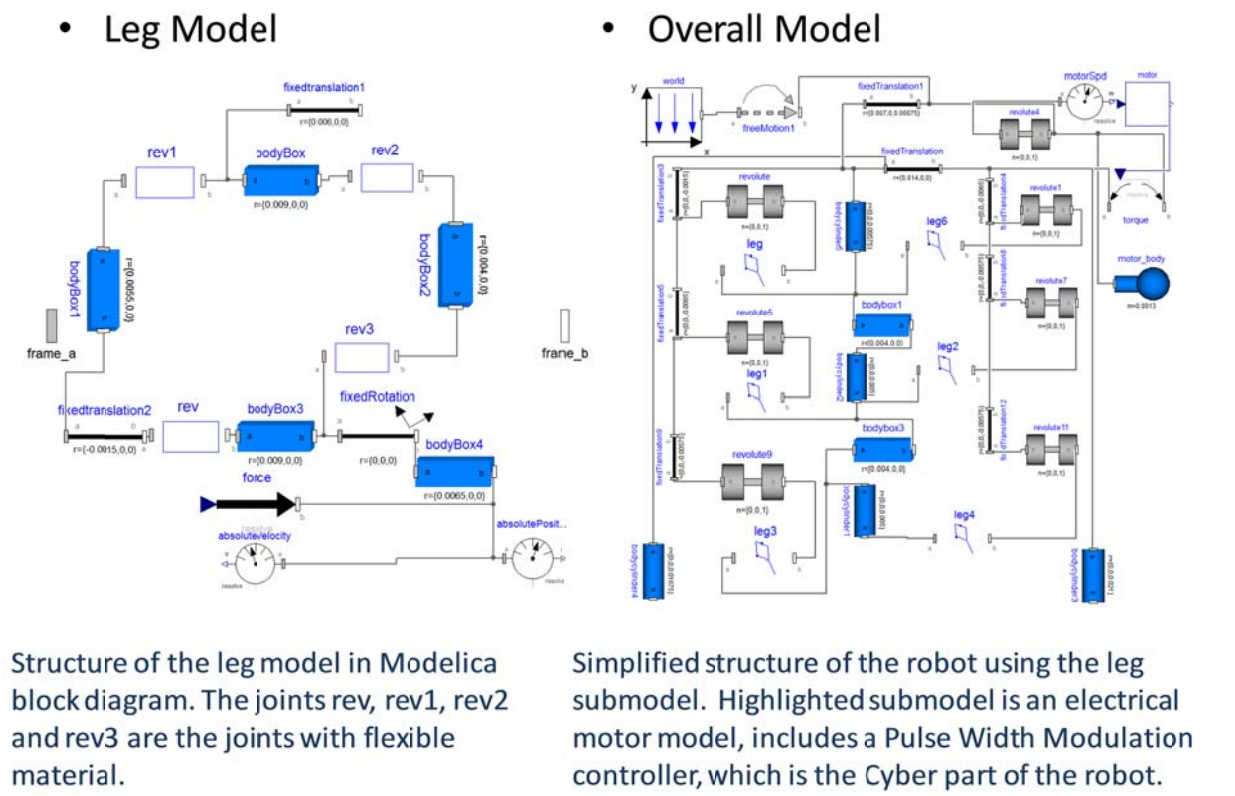


- Microrobots we are interested in are small insect-like robots with microfeatures, more specifically with flexible joints

- Real microrobot prototype on the left with Modelica DAE based model virtualized in Dymola on the right
- Dymola version has two distinct designs: (a) is the original design provided by Vogtmann-Gupta-Bergbreiter [VGB]

Fig. 13: Microrobots of interest as CPS

for applications related to collaborative motion such as, sensor networks, exploration and search-rescue in hazardous environments and medical drug delivery. However, control algorithms for these prototypes are very limited. Our new approach for modeling and simulation of the complete microrobotics system allows the robots to complete more complex tasks as per specifications. Since the microrobots tend to have small features, complex micro-structures and hierarchy, the control laws cannot be designed separately from the physical layer of the robots. Such a type of microrobot is indeed a CPS, as control in the cyber side, and the material properties and geometric structure in the physical side, are tightly interrelated. This design approach is important for microrobots, capable of collaborating and completing complex tasks. An important innovation in this work was the treatment of material properties and geometric configurations as design variables. Figure 14 illustrates a Modelica model of the microrobot. In this particular work we demonstrated how the interplay between the cyber part (here the control algorithm) and the physical part (here the material selection and the geometric configuration of the legs) can lead to various improved solutions with respect to stable motion of the microrobots. For instance, similar performance results could result with two different selections of the controller, material and geometry. Thus design space exploration also includes what part of functionality will be allocated to the physical part and what to the cyber and why?



Structure of the leg model in Modelica block diagram. The joints rev, rev1, rev2 and rev3 are the joints with flexible material.

Simplified structure of the robot using the leg submodel. Highlighted submodel is an electrical motor model, includes a Pulse Width Modulation controller, which is the Cyber part of the robot.

Fig. 14: Modelica model of microrobot

In [45] we developed HybridSim, a modeling and co-simulation toolchain for cyber-physical systems. Many CPS, such as Smart Buildings, are subject to very expensive deployment costs and complex network interactions. Thus comprehensive modeling and simulation of such systems are crucial to ensure that they function as intended before deployment. Given the multi-domain nature of CPS, it is more appropriate to use a heterogeneous simulation environment to study system dynamics. In [139], we designed and implemented an integrated modeling and co-simulation toolchain, called HybridSim, for the design and simulation of CPS. Firstly, HybridSim can transform and import existing system components from multi-domains into SysML, which enables systems engineers to design CPS with only these imported SysML blocks. Secondly, HybridSim can generate Functional Mock-up Units (FMUs) and configuration scripts directly from SysML designs. Finally, HybridSim can co-simulate these FMUs according to the Functional Mock-up Interface standard to synchronize their corresponding simulators and exchange information between them. We demonstrated the convenience and efficiency of HybridSim using a comprehensive hydronic heating system model for Smart Buildings as the case study to investigate the impact of packet loss and sampling rate introduced by the communication network.

A most successful application has been the creation of a synthesis environment for heterogeneous wireless sensor networks (WSN) [46, 47, 48, 49]. In our work heterogeneous wireless sensor networks (WSN), are viewed as large heterogeneous CPS. System design for Wireless Sensor Networks (WSNs) is a complicated process because of the wide variety of WSN applications, the heterogeneity of low-level implementation details, and the complex and heterogeneous interactions with their physical environments. Current ad hoc design methods (both simulation-based and testbed-based) for WSNs are far from satisfactory and cannot estimate system performance thoroughly and with the required accuracy. Furthermore, these ad hoc methods restrict design space exploration and the evaluation of new technology insertion. Existing ad hoc system design methods for Wireless Sensor Networks (WSNs) suffer from lack of reusability. In addition, the interactions between the continuous-time physical environments and WSNs have not been well studied.

.

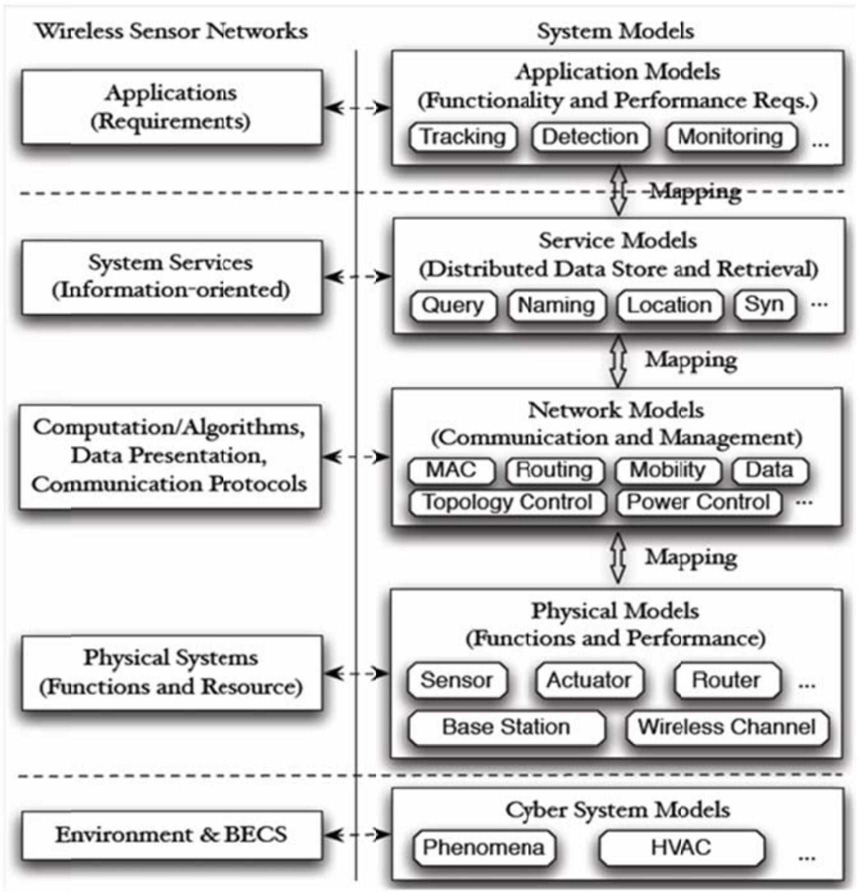

Wireless Sensor Networks
System Models
Applications (Requirements)
Application Models (Functionality and Performance Reqs.)
Tracking
Detection
Monitoring
...
Mapping
System Services (Information-oriented)
Service Models (Distributed Data Store and Retrieval)
Query
Naming
Location
Syn
...
Mapping
Computation/Algorithms, Data Presentation, Communication Protocols
Network Models (Communication and Management)
MAC
Routing
Mobility
Data
Topology Control
Power Control
...
Mapping
Physical Systems (Functions and Resource)
Physical Models (Functions and Performance)
Sensor
Actuator
Router
...
Base Station
Wireless Channel
Environment & BECS
Cyber System Models
Phenomena
HVAC
...

Fig. 15: WSNDesign model libraries

In our work [46, 48], we developed a model-based systems design (MBSD) framework for WSNs, called WSNDesign, which is a systematic methodology applying systems engineering principles to enhance model reusability and collaborations among multiple modeling domains. Firstly, WSNDesign provides model libraries (Figure 15) to model various behaviors and structures of WSNs in the context of Smart Buildings. Event-triggered components are either modeled in SysML Statechart Diagrams, or imported from existing TinyOS libraries. Continuous-time components are modeled in Modelica and their behaviors are described by differential equations, which are then transformed and imported to WSNDesign. Therefore, with the help of WSNDesign, system engineers can take advantage of many existing TinyOS and Modelica libraries, rather than design everything from scratch. Secondly, WSNDesign can estimate the performance of designed systems, providing instant feedback to sysstem engineers to quickly explore the performance trade-offs space. In WSNDesign, each component has a performance model described using SysML Parametric Diagrams. System overall performance can be calculated by traversing the system structure tree. Thirdly, WSNDesign can generate simulation codes and configuration scripts directly from system models. Although theoretical analysis provides immediate performance results, accuracy is often sacrificed due to oversimplifications, especially for large complex systems. With code generation, WSNDesign can save system engineers the trouble of writing simulation codes manually. WSNDesign integrates the existing widely accepted simulators to increase the confidence of the simulation results. Finally, WSNDesign provides an interactive tool to reduce the complexity of system analysis using summary propagation on factor graphs transformed from SysML Parametric Diagrams, and expose a sequence of design choices to system designers to provide instant feedback about the influence of a design decision on the complexity of system analysis. Figure 15 illustrates the model libraries of WSNDesign. Figure 16 shows the design flow of WSNDesign.

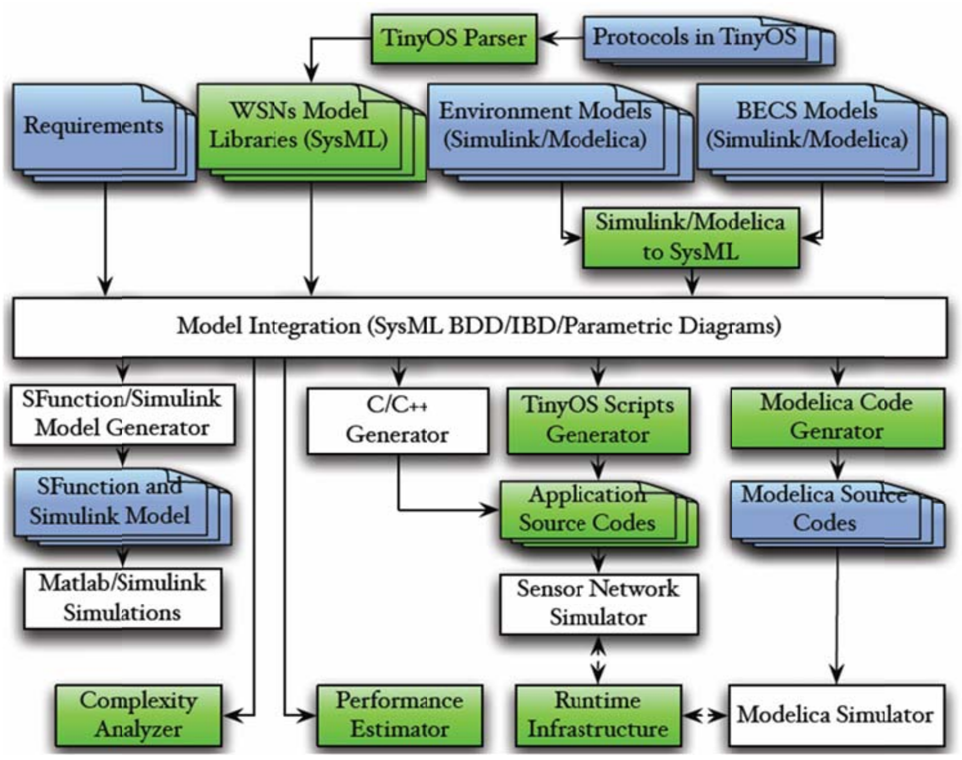

TinyOS Parser
Protocols in TinyOS
Requirements
WSNs Model Libraries (SysML)
Environment Models (Simulink/Modelica)
BECS Models (Simulink/Modelica)
Simulink/Modelica to SysML
Model Integration (SysML BDD/IBD/Parametric Diagrams)
SFunction/Simulink Model Generator
C/C++ Generator
TinyOS Scripts Generator
Modelica Code Genrator
SFunction and Simulink Model
Application Source Codes
Modelica Source Codes
Matlab/Simulink Simulations
Sensor Network Simulator
Complexity Analyzer
Performance Estimator
Runtime Infrastructure
Modelica Simulator

Fig. 16: WSNDesign design flow

On top of WSNDesign we designed and implemented HybridDB [49], an efficient distributed database system for flash-based storage-centricWSNs. HybridDB exploits a novel resource-aware data storage system, called HybridStore [47], to store and query sensor data in situ on each sensor mote. HybridStore can process typical joint queries involving both time windows and key value ranges as filter conditions extremely efficiently. Based on HybridStore, HybridDB provides the support for incremental ε-approximate querying that enables clients to retrieve a just-sufficient set of readings by issuing sub-queries with decreasing error-bounds. HybridDB will return an approximate dataset with arbitrary L^1-norm error bound, after applying temporal approximate locally on each sensor, and spatial approximate in the neighborhood on the proxy. In addition, HybridDB exploits an adaptive error distribution mechanism between temporal and spatial approximate for trade-offs of energy consumption between sensors and the proxy, and response times between the current subquery and following subqueries. Our implementation of HybridDB in TinyOS 2.1 can be transformed and imported to WSNDesign as a part of the model libraries.

8. Requirements Engineering Using Contract-based Design

The remaining last challenge (see (iv) in section 1) to add to the proposed Framework is a formal way to handle requirements. This means specifically a formal method to automatically annotate the structure and behavior components of the CPS by the mathematical representations of the specifications via constraints and metrics. This is currently done manually and as such it represents a scalability problem. As the complexity of the CPS increases, our inability to rigorously model the interactions between the physical and the cyber sides creates serious vulnerabilities. Systems become unsafe, with disastrous inexplicable failures that could not have been predicted. The challenges in the realization and operation of these CPS and systems of systems (SoS) are manifold, and cover a broad range of largely unsolved design and run-time problems. These include: modeling and abstraction, verification, validation and test, reliability and resiliency, multi-scale technology integration and mapping, power and energy, security, diagnostics, and run-time management. Failure to address these challenges in a cohesive and comprehensive way will most certainly delay if not prohibit the widespread adoption of these new technologies.

The most promising means to address this last challenge in MBSE of CPS is to employ structured and formal design methodologies that seamlessly and coherently combine the various dimensions of the design space (be it behavior, space or time), that provide the appropriate abstractions to manage the inherent complexity, and that can provide correct-by-construction implementations. The following technology issues must be addressed when developing new approaches to the design of complex systems, CPS and SoS [50]:

- The overall design flow for heterogeneous systems and the associated use of models across traditional boundaries are not well developed and understood. Relationships between different teams inside the same company, or between different stake-holders in the supplier chain, are not well supported by solid technical descriptions for the mutual obligations.
- **System requirement capture and analysis is in large part a heuristic process**, where the informal text and natural language-based techniques in use today are facing significant challenges. **Formal requirement engineering is in its infancy**: mathematical models, formal analysis techniques and links to system implementation must be developed.
- Dealing with variability, uncertainty, and life-cycle issues, such as extensibility of a product family, are not well addressed using available systems engineering methodology and tools.
- Design-space exploration is rarely performed adequately, yielding suboptimal designs where the architecture selection phase does not consider extensibility, re-usability, and fault tolerance to the extent that is needed to reduce cost, failure rates, and time-to-market.

- The verification and validation of "complex systems," particularly at the system integration phase, where any interactions are complicated and extremely costly to address, is a common need in defense, automotive, and other industries.

The challenge is to address the entire process and not to consider only point solutions of methodology, tools, and models that ease part of the design [50].

The proposed Framework for CPS MBSE addresses effectively these challenges with the exception of Requirements Engineering. The goal has been to offer a new approach to the system design problem, suited for the complexity and heterogeneity of CPS, that is rigorous and effective in dealing with the problems and challenges described above, and that, at the same time, does not require a radical change in the way industrial designers and manufacturers carry out their task as it cuts across design flows of different type.

Contract-based design [50, 51, 52] appears to be a promising methodology to address the remaining challenge, ***coupled with formal model-checking tools and methods like UPPAAL*** [53], ***efficient computation and approximation of reachable and invariant sets of set-valued hybrid systems*** [54] ***and automatic theorem proving tools and methods like Isabelle*** [55, 56].

Contracts in the layman use of the term are established when an OEM must agree with its suppliers on the subsystem or component to be delivered. Contracts involve a legal part binding the different parties and a technical annex that serves as a reference regarding the entity to be delivered by the supplier. Contracts can also be used through their technical annex in concurrent engineering, when different teams develop different subsystems or different aspects of a system within a same company.

In [5], it is argued that contracts can be actually used almost everywhere and at nearly all stages of system design, from early requirements capture, to embedded computing infrastructure and detailed design involving circuits and other hardware. Contracts [50, 51] explicitly handle pairs of properties, respectively representing the assumptions on the environment and the guarantees of the system under these assumptions. Intuitively, a contract is a pair $C = (A;G)$ of {Assumptions, Guarantees}, characterizing in a formal way 1) under which context the design is assumed to operate, and 2) what its obligations are. Assume/Guarantee reasoning has been known for quite some time, but it has been used mostly as verification mean for the design of software. The purpose in [50, 51, 52] is more ambitious: contract based design with explicit assumptions is a philosophy that should be followed all along the design, with all kinds of models, whenever necessary. The consideration of rich contracts as above in the industry is still in its infancy. To make contract-based design a technique of choice for system engineers, we must develop:

- Mathematical foundations for contract representation and requirement engineering that enable the design of frameworks and tools;
- A system engineering framework and associated methodologies and tool sets that focus on system requirement modeling, contract specification, and verification at multiple abstraction layers. The framework should address cross-boundary and cross-organizational design activities.

In [50] a unified treatment of contracts is provided, where they are precisely defined and characterized so that they can be used in design with no ambiguity. In addition, [50] provides an important link between interfaces and contracts to show similarities and correspondences.

UPPAAL [53] is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).

Isabelle [54] is a generic proof assistant. It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus

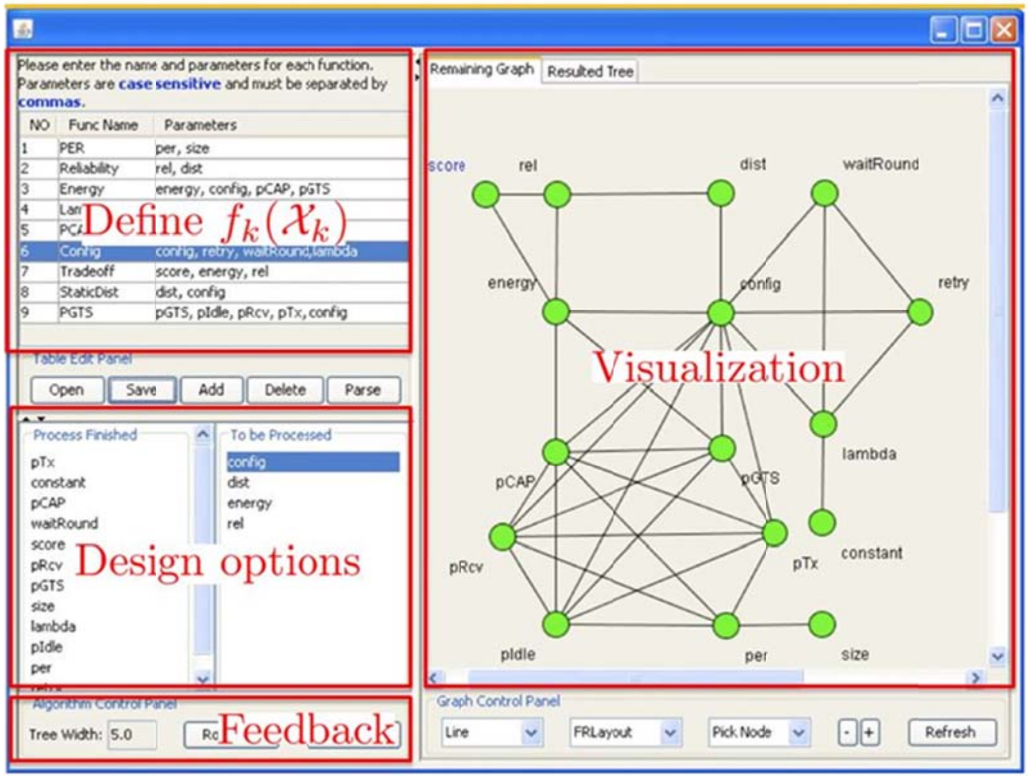
9. Compositional Analysis of Dynamic Networked CPS and Complexity Reduction

An important part of the proposed Framework for CPS is the development of methods and tools to manage the enormous complexity of these systems throughout their design and operations cycle. We have developed one such a method and tool in our research [57, 58] . Many more are needed.

Dynamic Bayesian networks (DBNs) can be effectively used to model various problems in CPS. In [57] we performed an empirical investigation on compositional analysis of DBNs using abstraction. In static systems and hidden Markov models, computation of a metric called treewidth induces a tree decomposition that can be used to perform logical or probabilistic inference and {max, +} optimizations in time exponential in treewidth and linear in overall system size. Intuitively, the linear scaling means that very large systems can be analyzed as long as they are sufficiently sparse and well structured. In these simple cases, summary propagation, which uses two operations, summation (projection) and product (composition), suffices to perform the inference or optimization. In this part of our research work, we extended this result to structured networks of communicating dynamic systems. We [57] defined generalizations of projection and composition operators that treat labeled Markov chains as primitive objects. The projection operation, corresponding to summation, is implemented as label deletion followed by exact state reduction for Markov chains, similar to Hopcroft's DFA minimization algorithm, with $O(n \log m)$ complexity. The composition operation is the product of state machines. We used canonical MDDs, similar to BDDs, to capture logical dependencies symbolically. The composition operation is the product of state machines. We used canonical MDDs, similar to BDDs, to capture logical dependencies symbolically. Combining symbolic representations with Markov chain lumping algorithms is a novel contribution. Using this approach, we have created a tool leveraging model based systems engineering technologies. The communicating Markov chains are specified using UML Statecharts via Papyrus extended using an ANTLR parsed domain specific language (DSL).

The tool reduces the number of states in networks of Markov chains by several orders of magnitude. In one example, a network having a product state space of more than 600 million states is reduced to about 500 states. A feature of this technique is that the state space is examined incrementally, meaning that the full state space is never explicitly represented, even as an input to the reduction algorithm. The primary reduction appears to come from symmetry which is surprising because the technique includes no explicit symmetry handling. We note that composition is efficient at least for systems with high symmetry. We have applied these methods and algorithms and tools to two CPSs: a modern aircraft power generation and distribution network and its management system (VMS), and a hospital intensive care unit (ICU).

We also developed an *Interactive Tree Decomposition Tool* for Reducing System Analysis Complexity [58]. The overall tool is based on a graphical tool for the calculation of *treewidth*, a metric on the parametric structure of a system that is intimately tied to the complexity of system analysis. For many graphically describable systems, such as systems of parametric equations, as in a SysML Parametric Diagram, Bayesian networks, mind maps, writing term papers, analysis of the system is exponential in treewidth and linear in system size. A tool facilitating comprehensive analysis can serve to bring competitive advantage to a systems engineering workflow by reducing costly unanticipated behaviors. Furthermore, a byproduct of computing treewidth is a framework for enumerating computationally compatible distributed algorithms.


Please enter the name and parameters for each function. Parameters are **case sensitive** and must be separated by **commas.**

NO	Func Name	Parameters
1	PER	per, size
2	Reliability	rel, dist
3	Energy	energy, config, pCAP, pGTS
4	Lan	
5	PCA	
6	Config	config, retry, waitRound,lambda
7	Tradeoff	score, energy, rel
8	StaticDist	dist, config
9	PGTS	pGTS, pIdle, pRcv, pTx, config

Tree Width: 5.0

Fig. 17: An illustration of our complexity analysis and reduction tool for CPS

Though there are classes for which treewidth computation is tractable (*chordal graphs*), it is generally NP-complete. For this reason, we pose [58] the problem from the perspective of finding satisficing solutions, exposing choices that can influence the complexity of the resulting system to the designer. A designer can contribute two important things to the structure of the system: a visual intuition about the relationships between the underlying objects and the ability to *change* the relationships themselves at design time to reduce analysis complexity. Having a visual tool that provides instant feedback will help designers achieve an intuitive grasp of the relationship between design decisions and system complexity. As complexity is the root of almost every systems engineering problem, and also something not easily understood, incorporating complexity analysis into a design process should improve resulting system designs.

Our tool [58] uses a randomized, anytime algorithm for interactive optimization of treewidth. It presents a sequence of choices to a designer and incrementally lowers an upper bound on system treewidth over time. This algorithm is novel, as few algorithms are targeted at interactivity with a human user. We have investigated a number of CPS examples for using the tool. We showed how our tool helps to decompose some example systems, including a quadrotor design optimization, a wireless sensor network design optimization, a Bayesian network, and a mind map. An instance of using the tool is illustrated in Figure 17.

References

[1] Special Issue, "Cyber-Physical Systems," *Proceedings of the IEEE*, January 2012, Vol. 100, No. 1.

[2] "Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0," *Final Report of the Industrie 4.0 Working Group, ACATECH*, German National Academy of Science and Engineering, federal Ministry of Education and Research, April 2013.

[3] J.S. Baras and M.A. Austin, "Second Semiannual Technical Progress Report, on Cooperative Agreement, NIST 70NANB11H148, *Modeling and Synthesis of Cyber-Physical Systems*," Institute for Systems Research, University of Maryland, November 18, 2012.

[4] V. Srinivasan et al, NIST CPS Architecture Task Group, "Architectures in the Context of Cyber-Physical Systems," *NIST Internal Report*, October 2012.

[5] D. Whitney et al, The ESD Architecture Committee, "The Influence of Architecture in Engineering Systems," *Engineering Systems Monograph*, MIT Engineering Systems Division, March 2004.

[6] "NIST Notional Reference Architecture for Cyber-Physical Systems," NIST Internal White Paper, April 2013.

[7] J.S. Baras and M.A. Austin, "Third Semiannual Technical Progress Report, on Cooperative Agreement, NIST 70NANB11H148, *Modeling and Synthesis of Cyber-Physical Systems*," Institute for Systems Research, University of Maryland, April 22, 2013.

[8] International Council on Systems Engineering (INCOSE): Systems Engineering Vision 2020. Version 2.03, TP-2004-004-02 (2007).

[9] J.S. Baras, *Lecture Notes for MSSE class, ENSE 621*, 2002.

[10] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML*, The MK/OMG Press, 2009.

[11] D. Spyropoulos and J.S. Baras, "Extending Design Capabilities of SysML with Tradeoff Analysis: Electrical Microgrid Case Study", *Proc. Conference on Systems Engineering Research (CSER'13)*, March 2013.

[12] *The eMoflon team: An Introduction to Metamodelling and Graph Transformations with eMoflon, V 1.4*, TU Darmsadt (2011).

[13] A. Anjorin, M. Lauder, S. Patzina, A. Schurr, "eMoflon: Leveraging EMF and Professional CASE Tools," *Proceedings INFORMATIK '11*, Bonn (2011).

[14] *No Magic,Inc.: Open API-User Guide. Version 17.0.1*, 2011.
[15] Meyer, J., Ball, M., Baras, J. S., Chowdhury, A., Lin, E., Nau, D., Rajamani, R., Trichur, V.: Process Planning in Microwave Module Production. In: Proc. SIGMAN: AI and Manufacturing: State of the Art and State of Practice (1998)
[16] Fan, M. K.H., Tits, A. L., Zhou, J., Wang, L.-S., Koninckx, J.: CONSOLE-User's Manual. Technical report, Un. of Maryland, Vers. 1.1 (1990)
[17] Fan, M. K.H., Wang, L.-S., Koninckx, J., Tits, A. L.: Software Package for Optimization-Based Design with User-Supplied Simulators. IEEE Control Systems Magazine, Volume 9, Issue 1, Pages 66 - 71 (1989)
[18] Tischler, M.B., Colbourne, J.D., Morel, M.R., Biezad, D.J.: A Multidisciplinary Flight Control Development Environment and its Application to a Helicopter, IEEE Control Systems Magazine, Volume 19, Issue 4, Pages 22-33 (1999)
[19] Potter, P.J.: "Parametrically Optimal Control for the UH-60A (Black Hawk) Rotorcraft in Forward Flight," *MS Thesis*, Un. of Maryland, 1995.
[20] Bradwell, R., Ford, J., Mills, P., Tsang, E.P.K. & Williams, R, "An overview of the CACP project: modelling and solving constraint satisfaction/optimization problems with minimal expert intervention," *Workshop on Analysis and Visualization of Constraint Programs and Solvers, Constraint Programming 2000*, Singapore 22 September 2000.
[21] E. Tsang, "A Glimpse of Constraint Satisfaction," *Artif. Intell. Rev.* 13, 3, 215-227, June 1999.
[22] J-F. Puget, "Applications of Constraint Programming," in U. Montanari and F. Rossi, (ed.), *Proceedings, Principles and Practice of Constraint Programming* (*CP'95*), 647-650, Lecture Notes in Computer Science, Springer, 1995.
[23] H. Simonis, "The CHIP System and its Applications," in U. Montanari, and F. Rossi, (ed.), *Proceedings, Principles and Practice of Constraint Programming* (*CP'95*), 643-646, Lecture Notes in Computer Science, Springer, 1995.
[24] J. Lever, M. Wallace, and B. Richards, "Constraint Logic Programming for Scheduling and Planning," *British Telecom Technology J.*, Vol.13, 1., 73-80, Martlesham Heath, 1995.
[25] A. Colmerauer, "An Introduction to Prolog III," CACM Vol.33, No7, 69-90, July 1990.
[26] K. Marriott and P. J. Stuckey, *Programming with Constraints, An Introduction*, MIT Press, 1998.
[27] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & sons, 1998.
[28] D. N. Bertsekas, *Nonlinear Programming*, 2nd Edition, Athena Scientific, 1999.
[29] T. Muller, "Constraint Propagation in Mozart," *Doctoral Dissertation*, U. Saarbrucken, 2001.
[30] T. Bapty, S. Neema, J. Scott, J. Sztipanovits, S. Asaad, "Model-Integrated Tools for the Design of Dynamically Reconfigurable Systems," VLSI Design, vol. 10, 281-306, 2000.
[31] C. Van Buskirk, B. Dawant, G. Karsai, J. Sprinkle, G. Szokoli, K. Suwanmongkol, R. Currer, "Computer-aided Aircraft Maintenance Scheduling," *ISIS-02-303*, Institute for Software Integrated Systems, November, 2002.
[32] S. Bistarelli, U. Montanari, and F. Rossi, "Soft Concurrent Constraint Programming," *ACM Trans. Comput. Logic*, 7(3), 563-589, 2006.
[33] K. Apt, *Principles of Constraint Programming*, Cambridge University Press, 2003.
[34] T. Elrad, R. E. Filman, A. Bader, "Aspect-oriented programming: Introduction", *Commun. ACM* 44, 10 (Oct. 2001), 29-32.
[35] https://www.fmi-standard.org

[36] M. Otter, H. Elmqvist, T. Blochwitz, J. Mauss, A. Junghanns, H. Olsson, “Functional Mockup Interface – Overview” (http://synchronics.inria.fr/lib/exe/fetch.php/modelica-fmielmqvist.pdf). http://synchronics.inria.fr (INRIA), 2011.

[37] J. Bastian, C. Clauss, S. Wolf, and P. Schneider, “Master for Co-Simulation Using FMI,” Proc. 8th International Modelica Conference, 115-120, Dresden, Germany, 2011.

[38] J. G. Michopoulos, C. Farhat, and J. Fish, “Modeling and Simulation of Multiphysics Systems,” *Transactions of the ASME, J. of Computing and information Science and Engineering*, September 2005, Vol. 5, 198-213.

[39] D. Keyes et al, “Multiphysics Simulations: Challenges and Opportunities,” *Technical Report ANL/MCS-TM-321*, Argonne National laboratory, 2011.

[40] Modelica and the Modelica Association, https://www.modelica.org/.

[41] Dymola, Dynamic Modeling Laboratory, http://www.3ds.com/products-services/catia/capabilities/catia-systems-engineering/modelica-systems-simulation/dymola/.

[42] *COMSOL MULTIPHYSICS*, *Product Booklet*, 2013, www.comsol.com.

[43] “Multiphysics Simulation,” *IEEE Spectrum*, May 2013.

[44] Y. Zhou and J. S. Baras, “CPS Modeling Integration Hub and Design Space Exploration with Application to Microrobotics,” in D. C. Tarraf (ed.), *Control of Cyber-Physical Systems*, 23-42, LNCIS 449, Springer 2013.

[45] B. Wang and J.S. Baras, “HybridSim: A Modeling and Co-simulation Toolchain for Cyber-Physical Systems,” *Proceedings 17th IEEE/ACM DS-RT Conference*, October 30, 2013.

[46] B. Wang and J. S. Baras, “Integrated Modeling and Simulation Framework for Wireless Sensor Networks,” *Proceedings of the 21st IEEE International Conference on Collaboration Technologies and Infrastructures (WETICE 2012- CoMetS track)*, pp. 268-273, Toulouse, France, June 25 - 27, 2012.

[47] B. Wang and J. S. Baras, “HybridStore: An Efficient DataManagement System for Hybrid Flash-based Sensor Devices,” *Proceedings of the 10th European Conference on Wireless Sensor Networks*, pp. 50-66, Ghent, Belgium, February 13-15, 2013.

[48] B. Wang, “Storage-centric Sensor Networks for Smart Buildings,” *Proceedings of the 12th ACM/IEEE Conference on Information Processing in Sensor Networks IPSN 2013,* Philadelphia, PA, April 8-11, 2013.

[49] B. Wang and J. S. Baras. “HybridDB: An Efficient Database System Supporting Incremental ε-Approximate Querying for Storage-centric Sensor Networks,” submitted to *ACM Transactions on Sensor Networks*, May 2013.

[50] A. Benveniste et al, “Contracts for Systems Design,” *INRIA Research Report No 8147*, Nov. 2012.

[51] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, “Taming Dr. Frankenstein; Contract-Based Design for Cyber-physical Systems,” *European J. Control*, Vol. 18, N. 3, 217-238, 2012.

[52] A. Benveniste, “Contracts and Interfaces in the Context of Requirements Engineerng,” invited address, ICECCS 2012, Paris, France, July 2012.

[53] UPPAAL, http://www.uppaal.org/.

[54] P. Collins, “Optimal Semicomputable Approximations to Reachable and Invariant Sets,” *Theory of Computer Systems*, 2007, DOI: 10.1007/s00224-006-1338-3.

[55] Isabelle, http://www.cl.cam.ac.uk/research/hvg/Isabelle/.

[56] T. Nipkow, L.C. Paulson and M. Wenzel, *A Proof Assistant for Higher-Order Logic*, Springer-Verlag, 2013.

[57] S. Yang, Y. Zhou, J.S. Baras, “Compositional Analysis of Dynamic Bayesian Networks and Applications to Complex Dynamic System Decomposition,” *Proceedings of the Conference on Systems Engineering Research (CSER’13)*, Atlanta, GA, March 19-22, 2013.

[58] S. Yang, B. Wang and J.S. Baras, “Interactive Tree Decomposition Tool for Reducing System Analysis Complexity,” *Proceedings of the Conference on Systems Engineering Research (CSER’13)*, pp. 167-176, Atlanta, GA, March 19-22, 2013.