Foundations of Hybrid and Embedded Software Systems

Principal Investigator

- Shankar Sastry (UC Berkeley, EECS)

Co-Principal Investigators

- Thomas Henzinger (UC Berkeley, EECS)
- Edward Lee (UC Berkeley, EECS)
- Alberto Sangiovanni-Vincentelli (UC Berkeley, EECS)
- Janos Sztipanovits (Vanderbilt, Electrical and Computer Engineering)

Senior Investigators

- Bella Bollobas (University of Memphis, Mathematics)
- Alex Aiken, Electrical Engineering and Computer Sciences (UC Berkeley, EECS)
- Ruzena Bajcsy (UC Berkeley, EECS)
- Gautam Biswas (Vanderbilt, Computer Sciences)
- David Culler (UC Berkeley, EECS)
- Kenneth Frampton (Vanderbilt, Mechanical Engineering)
- Karl Hedrick, (UC Berkeley, Mechanical Engineering)
- Gabor Karsai (Vanderbilt, Electrical and Computer Engineering)
- Kurt Keutzer (UC Berkeley, EECS)
- David Messerschmitt (UC Berkeley, EECS)
- George Necula (UC Berkeley, EECS)
- Satinderpaul S. Devgan (Tennessee State, Electrical Engineering & Computer Science)
- Pravin Varaiya (UC Berkeley, EECS)

International Collaborators

- Albert Benveniste (IRISA INRIA, Rennes, France)
- Hermann Kopetz (Technical University of Vienna, Austria)
- Manfred Morari (ETH, Zurich, Switzerland)
- Joseph Sifakis (CNRS VERIMAG, Grenoble, France)
- Kim Larsen (University of Aalborg, Aalborg, Denmark)
- Henrik Christensen (Royal Institute of Technology, Stockholm, Sweden)

Table of Contents

Project Summary

This ITR project is aimed at developing the foundations of a modern systems science that is simultaneously computational and physical; it remarries time, concurrency, robustness, continuums, and resource management to computation. The lead partners in the proposed consortium are the Center for Hybrid and Embedded Software Systems (CHESS) at the University of California at Berkeley (UCB), the Institute for Software Integrated Systems (ISIS) at Vanderbilt (VU), and the department of Mathematical Sciences at the University of Memphis (UM). This new systems science represents a major departure from the current, separated structure of computer science (CS), computer engineering (CE), and electrical engineering (EE): it reintegrates information and physical sciences.

Educational Impact. The impact of this change on teaching and research is profound, and cannot be confined to the graduate level. Based on the ongoing, groundbreaking effort at UCB, we propose to deliberately re-architect and retool undergraduate teaching at the participating institutions, and to engage in course development at a set of California community colleges with which UCB has established relationships and which have a high enrollment of Hispanic and African American students. We will also engage undergraduate students from around the country via the UCB SUPERB program, an established NSF REU site, and the VU SURGE program, which have focused on preparing highly successful students of color to pursue advanced degrees. At the participating institutions, we will replace the 30-year-old conventional curriculum in systems science with one that admits computation as a primary concept. The curriculum changes will be aggressively promoted through a process of workshops and textbook preparation, to adapt the technical material to the needs of the participating institutions and will be disseminated widely. Faculty and graduate student researchers from minority and other institutions will be recruited each summer to participate in a program called SIPHER (Summer Internship Program in Hybrid and Embedded Software Research).

Research (Technical Impact). The proposed ITR has four focus areas of research. (a) *Hybrid systems theory*. The focus here is on scaling up pioneering approaches that integrate physical modeling with computational systems. Existing methods work for simple, low-dimensional systems; we will find methods that apply to complex, interconnected systems with stochastic attributes. As part of this effort, the mathematical foundations of systems theory need to be rebuilt in a way that tightly integrates continuous and discrete domains. (b) *Model-based design*. The main effort here is to develop a set of models with solid mathematical foundations that allow for the systematic integration of diverse efforts in system specification, design, synthesis, analysis and validation, execution, and design evolution. Key to this effort is research in concurrent and real-time models of computation, and in meta-modeling and meta-languages for composing, transforming, and validating domain-specific models. (c) *Advanced tool architectures*. The deliverables from this project will be a set of reusable, inter-operating software modules, freely distributed as open-source software. These modules will be toolkits and frameworks that support the design of embedded systems, provide infrastructure for domain-specific tools, and provide model-based code generators. (d) *Experimental research*. The program will leverage existing system-building efforts involving avionics, anti-terrorism technologies, vehicle electronics, and autonomous robots. In addition we will apply our methods to networks of embedded systems for applications such as environment monitoring, building protection, and emergency response. Embedded systems laboratories will be established at the partner institutions to host experimental applications of these results. The experimental validation effort will provide the community with case studies for the toolkits and infrastructure developed in this program.

Research (Broader Societal Impact). This project, because of its focus on foundations, will provide a fundamentally new paradigm, based on hybrid systems, for modeling and analysis of many complex phenomena that occur in the physical and biological sciences on both microscopic and macroscopic levels. Because of its attention to methodologies, our research will generate new paradigms and tools for the design of complex embedded systems to ensure reliability, performance, low power consumption and cost beyond what can be achieved by current, *ad hoc* practices. These outcomes of the project are prerequisites for the deployment of embedded, autonomous computing in many safety-critical applications, from medical devices to transportation to national security needs in avionics. Finally, the attention to a new education model will create a new generation of engineers who will be able to master the design of complex, heterogeneous systems that will be the backbone of the future IT industry.

Project Description

The science of computation has systematically abstracted away the physical world. The science of physical systems has systematically ignored computational limitations. Embedded software systems, however, engage the physical world in a computational manner. We believe that it is time to construct a Modern Systems Science (MSS) that is simultaneously computational and physical. Time, concurrency, robustness, continuums, and resource management must be remarried to computation.

At UC Berkeley (UCB), the Center for Hybrid and Embedded Software Systems (CHESS) was founded with the explicit mission to build and disseminate MSS. At Vanderbilt University (VU), the Institute for Software Integrated Systems (ISIS) is the leading proponent of model-integrated computing, a paradigm that is central to MSS. At the University of Memphis (UM), the Mathematical Sciences Department conducts groundbreaking research on phase transitions in computational complexity, which has fundamental importance in dynamic, embedded computing applications.

We propose a program that includes the long-term, high-risk, high-reward, basic scientific research necessary to build the foundations of MSS, and a sustained effort to create a new generation of engineers that are comfortable with the juncture of computation and physical phenomena. The research will be carried out by UCB-CHESS, VU-ISIS, and UM. Educational outreach programs will include the California community college system, which feeds many of the engineering students to UCB and other State Universities, and HBCUs and universities with high minority populations in the South. The proposal to the NSF-ITR has the potential of high leverage from other activities of the participating organizations paid for by other means, such as university and state investment and industry funding.

1. Research Rationale

The fusion of information processing with physical processes changes the physical world around us. From toys to aircraft and from cars to factory robots, computers monitor and control our physical environment. Information processing that is tightly integrated with physical processes is called *embedded computing*. Embedded computing is becoming the universal system integrator for physical systems. Its pervasiveness is well illustrated by the following facts: (a) the total shipment of microprocessor units and micro control units in 1997 was over 4.4 billion units, and of this about 98% related to embedded applications [94]; and (b) between 1994 and 2004 the need for embedded software developers is expected to increase 10-fold [28].

Embedded software development is one of the grand challenges in computer science today. Many of the abstractions that have been so effective at improving our computational capabilities are either indifferent to or at odds with the requirements of embedded software. In fact, embedded software programming has never transitioned from a resource-driven (architecture-centric) to a requirements-driven (application-centric) paradigm. The result is costly, over-engineered, brittle systems. The generous attitude of desktop software vendors towards system reliability is not acceptable here—for example, failures are intolerable in safety-critical systems such as vehicle electronics, avionics, and transportation systems. And over-engineering (exaggerated safety margins) is unacceptable in systems that use expensive or limited physical resources.

We believe that in order to address these issues, a modern systems theory that provides a solid mathematical foundation for embedded software design is needed. While traditional systems theory focuses on behavioral attributes of small, homogeneous, idealized systems (for example, linear time-invariant systems), a modern systems theory must be based upon attributes of computational systems such as concurrency, hierarchy, heterogeneity, resource awareness, and controlled complexity. Groundbreaking research in these areas has been carried out at UCB, VU, and UM. We believe that the time is right to integrate and focus these efforts to construct a modern systems science for embedded software systems, and to transition the results under the umbrella of a multi-university effort.

The proposed research has major potential impact in engineering and the sciences. Embedded computing controls physical interactions in engineered systems, that is, energy and material flows that have direct and immediate impact on the physical environment and the people involved. As a result, almost all embedded software is subject to high or extremely high assurance requirements [48]. Increasingly, the safety and dependability of our cars, airplanes, weapon systems and critical infrastructure monitoring systems depend on the foundations of our embedded system technology, which is the central objective of our research on MSS.

Moreover, while the development of MSS will be driven primarily by the need for scalable techniques in embedded systems engineering, any fundamental advance in the understanding of processes that are both physical and computational in nature will have ramifications far beyond embedded software design. For example, at the microscopic level, the traditional abstractions of computer science will break at the level of future molecular computing devices, whose physicality will be used in computation in ways that are far more intricate than the Boolean state abstraction used for transistors. Second, at the macroscopic level, a very large network of interacting computational elements gives rise to emergent behavior that will be best understood in continuous, physical terms, in the same way in which the behavior of a gas is best understood as a continuous entity rather than a discrete collection of individual molecules. In turn, the continuous behavior of such an entity may undergo mode transitions, which is again a discrete phenomenon. Such observations have led recently, for example, to the application of hybrid systems theory in areas as diverse as the design of asynchronous circuits [59] and the modeling of biological systems [10]. We are convinced that the time is right to make a focused effort in establishing MSS and developing a new understanding of its implications on system modeling and analysis, design technology, and tool architectures in a carefully selected set of representative applications.

1. Research Focus Areas

The proposed research program has four focus areas: hybrid systems theory, model-based design, advanced tool architectures, and experimental research. *Hybrid systems theory* will build the mathematical foundation of MSS. This foundation needs to be grounded both in the continuous mathematics of physical processes and the discrete mathematics of computational processes. *Model-based design* will build a scalable methodology for systems design and analysis based on hybrid systems theory. Model-based design controls complexity by supporting the manipulation and integration of models for multiple design aspects. *Advanced tool architectures* will provide software support for model-based design. An open software infrastructure will accommodate design and analysis tools as inter-operating components. *Experimental research* will guide the theory and tool development. Special emphasis will be on applications with societal impact, such as networked embedded systems for environmental monitoring and embedded control systems that address national and homeland security needs. The overarching theme in our research is *compositionality*. We will pursue compositionality in hybrid system theory, we will use composable models and model manipulation methods in model-based design, and we will investigate composable tool architectures that enable the rapid integration of domain-specific design environments. Compositionality enables the separation of orthogonal concerns, and the integration and reuse of solutions (theories, models, tools), and thus makes the significant objectives of this project feasible.

1.1 Hybrid Systems Theory

A *hybrid system* is a system that contains both physical and computational processes. A typical example of a hybrid system is embedded control software, which interacts with a physical environment through sensors and actuators. Hybrid systems theory, therefore, lies at the foundation of any effort to systematically build a modern systems science (MSS). So far, hybrid systems theory has been developed, largely at UCB, only for small, idealized systems, such as *hybrid automata* [12][65]. A scalable MSS must address the following key issues. *Key faculty participation here will be from Biswas, Bollobas, Henzinger, Lee, Sangiovanni-Vincentelli, Sastry, Sztipanovits, and Varaiya.* A graduate text is in preparation in this area based on a course being taught at UCB [125].

1.1.1 Deep compositionality

In the early 90s, a realization began to set in that, on the one hand, systems modeling techniques from classical electrical (systems) engineering are inadequate for capturing the computational aspects of systems implemented increasingly in software, and on the other hand, the computational models from classical computer science are inadequate for capturing the physical aspects of software that interacts with physical processes. The new field of *hybrid systems* was founded on the premise that both bodies of knowledge need to be combined for the design and analysis of embedded software. There are two ways in which such a combination may be achieved. In a *shallow* combination, hybrid systems are described in a language that results from connecting expressions describing physical processes (such as difference and differential equations) with expressions describing computational processes (such as state machines or pseudo-code). While a shallow combination enables the description (requirements specification, architectural and behavioral description) of mixed physical-computational systems, it does not, *per se*, support the design and analysis of such systems. For this, we need a *deep* combination of the two worlds. Deep compositionality requires that the properties of a composite system can be derived solely from the properties of the component systems and the type of the connection. In the case of hybrid systems, this must apply to both computational properties—functionality, efficiency, accuracy—and physical properties—stability, timing, resource usage.

Compositionality of computational systems has long been studied in computer science, both in the case of *sequential* connections such as procedure calls (semi-formally in software engineering, and formally in programming language semantics [174][132]), and in the case of *concurrent* connections (semi-formally in hardware description languages, and formally in concurrency theory [93][131][126][42] [103][62]). The mathematics of compositionality is a discrete one, based on logic, combinatorics, and universal algebra (fixed points). Properties of computational systems become interesting through complex patterns of composition, such as iteration and recursion, concurrency, inheritance, and dynamic process creation. In contrast, the properties of physical systems (stability, for example) can be nontrivial even in simple cases, and much of the focus in systems theory has been on standardized system architectures, such as feedback loops. The mathematics of physical properties is a metric one, based on linear algebra, topology, and analysis. It is the underlying fundamental dichotomy between Boolean (true, false) and metric (quantitative) data types that separates the two disciplines on all levels. Obtaining a theory of deep compositionality for hybrid systems requires a complete rebuilding of the mathematical foundation, based on a core collection of new concepts that combine or replace the main concepts of systems theory and computation theory [125][136][138].

Preliminary progress in this direction [23][89][127][171] still falls short of the proposed science of deep compositionality. For example, efforts such as Masaccio [67] and Charon [16] do not address resource and stochastic performance properties, nor do they achieve the kind of scalability needed in practice, lacking essential features such as model parameterization, instantiation with inheritance, and dynamic model reconfiguration. To control the complexity inherent in combining this multitude of issues, we are pursuing a systematic attempt to orthogonalize concerns. For example, we are studying ways to add, say, model instantiation to *any* deeply compositional formalism in a way that preserves deep compositionality with respect to the properties of interest. In other words, we are striving to develop the mathematics for a meta-theory of models of computation, with deep compositionality being one of the meta-properties. Just as on the level of a particular formalism, properties such as safety should be preserved by the putting-together of systems, on the meta-level, properties such as compositionality should be preserved by the putting-together of formalisms. It is our hope that in this way we can also arrive at a deeply compositional theory of *stochastic* hybrid systems, which has proved elusive despite repeated attempts [8][97][137]. Deep compositionality can be exploited in system design through the use of component interfaces, and in system analysis through the use of assume-guarantee reasoning.

Interfaces for hybrid components. The notion of *interface* is central to the composition and decomposition of systems. An interface must expose sufficient information about a system for determining if two systems are *compatible*, but it should expose no more than that. In component-based design, interfaces facilitate the bottom-up assembly of a system by providing a compatibility check for components that originate from different vendors or libraries. Dually, interfaces also facilitate the top-down design of a system by providing contracts that can be handed to the independent designers of individual components and guarantee that if each component implementation satisfies its interface contract, then the overall, composite system is well-formed and meets its requirements. A paradigmatic class of software interfaces is *types*. Unlike specification formalisms such as temporal logics, types specify not only output behavior, but also constrain input behavior: the environment, when interacting with the component, is expected to respect its type; otherwise, the compiler will signal a violation of this expectation. While traditional type systems specify static requirements on input and output values, recently researchers have begun to develop type systems for dynamic requirements [51][40][111][6][119], such as the ordering of method calls. We have formalized such generalized, behavioral type systems as *interface theories* [7]. For example, a suitable interface theory permits us to specify, and check, that the *read* method of a file server must not be called before the *open* method. Currently we are developing an interface theory for timing constraints, which permit us to specify and check, for instance, that a software task completes before a deadline provided that the frequency of external interrupts is bounded. For hybrid components, we will need to consider more general physical and computational constraints, such as stochastic behavior, resource usage, stability, and convergence.

Assume-guarantee reasoning for hybrid systems. In the analysis of component-based systems, as in design, it is usually insufficient to consider components in isolation; rather, one must consider each component together with an assumption about the environment in which the component is placed. In other words, a component typically meets its requirements only if the environment does likewise. Such reasoning is inherently circular, and not always sound. For example, the circular ("assume-guarantee") reasoning is sound for safety properties: if component A won't fail (the guarantee) provided B does not fail (the assumption), and B won't fail provided A does not fail, then the composition of A and B will never fail [1][128][19]. The same reasoning, however, is unsound for liveness properties: if A will respond eventually provided B does, and B will respond eventually provided A does, this does not ensure that the composition of A and B will ever respond. Little is known about assume-guarantee reasoning for richer properties of the kind we are interested in, such as timing, resource usage, and performance [85][8]. Indeed, such properties are inherently non-compositional if components are viewed in isolation; for example, if two components share the same resource, then their individual worst-case timing properties do not apply to the composition. Therefore, assume-guarantee reasoning, with its emphasis on environment assumptions, is essential for composing physical and computational properties.

1.1.2 Robust hybrid systems

Current models of hybrid systems have the problem that they are, in a sense, too "exact." Consider, for example, a *timed automaton* [14], which is a finite automaton equipped with real-valued variables (so-called "clocks") which measure the progress of time. The transitions of such an automaton can be guarded by conditions on the clocks. In particular, one transition may be enabled when $x \neq 5$, for a clock x, and another one, when $x=5$. This situation, where the transition time must be determined with "infinite precision," does not represent a physical process, which typically can be modeled only approximately using error terms, nor can it be implemented as a computational process, which has limited space and time resources. We have made preliminary attempts to develop more "robust" models of hybrid systems, by considering not individual behaviors over time, but "bundles" of behaviors that are proximate in an appropriate topology [61]. These attempts, however, have largely been unsuccessful, as they add a layer of complexity to the traditional, already brittle, models. It has become our sense that we must instead look for a way of redefining fundamental concepts such as *behavior* and *property* in a way that is less fragile. This can be done by borrowing ideas from economics, in particular, *discounting* of the future.

Suppose we redefine the value of a property at a state to be not 0 (false) or 1 (true), but a real value in the interval [0,1]. The value is computed using a discount factor $d \in [0,1]$: if the property is violated in the current state, then its value is 0; if the property is fulfilled in the current state, then its value is 1; otherwise its value is d times the minimal (or maximal, depending on the "polarity" of the property) value of the property at successor states. In this way, the value of a safety property is inversely determined by the length of the shortest unsafe trajectory. As the discount factor d goes to 1, we obtain the classical, boolean value of the property. This view extends naturally from purely discrete to stochastic, timed, and hybrid systems. Moreover, in these systems the real value of a property often has a natural "physical" interpretation, such as probability, time, or cost. We can capture, for example, optimization criteria and performance characteristics. The redefinition of properties in this way is, by itself, not sufficient to obtain robust models—the introduction or removal of a single transition can still cause an arbitrary, unbounded change in the value of a property. However, we can similarly define a topology on systems, where proximity is based on a *discounted* notion of equivalence. For example, the discounted bisimilarity of two states is a real number between 0 (not at all bisimilar, i.e., distinguishable at present) and 1 (perfectly bisimilar, i.e., indistinguishable in all future) [41]. Then we can show that if two states have a high bisimilarity value, then any given property has almost the same value in both states. In this way, we obtain a continuous (epsilon-delta) view of stochastic hybrid systems [97]: if such a system is perturbed in a small way (say, by changing the timing or probability of a transition), then the values of its properties change only proportionally. The hope that the fundamental notions from concurrency modeling such as bisimulation can be redefined in a way that gives rise to a topology of hybrid systems is indeed the hope that there is a way to fully and elegantly bridge CS and EE systems theory. It hints at how we may be able to inject, uniformly and systematically, continuous "physicality" into discrete computational systems. The reverse problem, of injecting computational issues into physical processes, is addressed next.

1.1.3 Computational hybrid systems

We will investigate hybrid systems from the numerical and computational complexity points of view. For the computer simulation and execution of hybrid systems, computational limitations render idealized mathematical models inadequate. Issues such as event detection and Zeno behavior need to be addressed, and a theory of error estimation for hybrid systems needs to be developed [109][141]. We will address these problems by pursuing a constructive, computational approach to hybrid system design, and to controller synthesis in particular. In the continuous case, optimal control laws may be derived as solutions to the Hamilton-Jacobi-Bellman equation, while discrete controllers can be synthesized by solving games on finite automata. Both methods can be seen as special cases of a generic game-theoretic approach [167]. To convert this general approach into a constructive synthesis procedure, we need efficient numerical tools to compute the solutions of Hamilton-Jacobi equations. Approximation techniques are a first step, where we will use ellipsoid, linear hyperplane, and fast wavefront methods [166][173][108]. In addition, we will use quantitative computing methods when the equation has shocks, corresponding to changes in the gaming strategy. For systems with high-dimensional state spaces or for many agents, a hierarchical application of the approach facilitates least-restrictive control computations [107]. We will also investigate the projection of control objectives onto the state spaces of individual agents, and the use of systematic decomposition techniques based on the data provisioning paths of the underlying architecture [153].

1.1.4 Phase transitions

Search-intensive algorithms are essential to the analysis of hybrid systems. Resource allocation, scheduling, planning, and combinatorial optimization are necessary for implementing new capabilities in a variety of emerging applications. The fundamental challenge is that search-intensive algorithms can easily lead to computationally intractable problem instances. Research in average- or typical-case complexity has recently led to the recognition that the hardness of a random instance of a problem may be related to the phase transition that the randomly constrained system undergoes. To be more precise, it has been observed that one can define control parameters in terms of which search-intensive problems undergo phase transitions. The study of phase transitions in large combinatorial systems has brought together statistical physics, discrete mathematics, probability theory, and theoretical computer science. This combination of techniques has led to much recent progress in the study of the "random k-SAT problem," which studies satisfiability of typical instances of a fixed number k of clauses [133][38][110][106][54]. The best result to date is [27]: they determined the exact scaling window of the phase transition for 2-SAT. This result was much sharper than anybody had hoped to prove. The expertise we have gained

in techniques of statistical physics, combinatorics, and computer science in solving this problem should enable us to tackle the more complex phase transition problems that arise in hybrid systems.

1.2 Model-based Design

While hybrid systems theory provides a semantic, mathematical foundation for the integrated modeling of physical and information systems, model-based design focuses on the formal representation, composition, and manipulation of models during the design process. It addresses system specification, model transformation, synthesis of implementations, model analysis and validation, execution, and design evolution. The semantic frameworks in which these models are applied may be domain-specific, offering embedded system designers methods and syntaxes that are closer to their application domain. To do this well, they must emphasize concurrency, communication abstractions, and temporal properties, rather than procedural interfaces. For example, domain-specific semantic frameworks for embedded systems might represent physical processes using ordinary differential equations, signal processing using dataflow models, decision logic using finite-state machines, and resource management using synchronous models. *Key faculty involved are Aiken, Henzinger, Karsai, Keutzer, Lee, Necula, Sangiovanni-Vincentelli and Sztipanovits*.

One approach to model-based design, called *actor-oriented* design [115][122], has been extensively researched at UCB. The term *actor-oriented* refers to a refactored software architecture, where instead of objects, components are parameterized actors with ports. A port represents an interaction with other actors; its precise semantics is determined by a *model of computation*, which captures the interaction and coordination between actors. (Although the term "actor" is often associated with Agha [3], the UCB use of the term is broader, in that actors are not required to encapsulate a thread of control.) Many models of computations have been developed, including the domain-specific frameworks listed above.

There are many other examples of actor-oriented frameworks, including Simulink (from The MathWorks), LabVIEW (from National Instruments), and Cocentric System studio (from Synopsys), all of which are used for embedded systems design. The approach has not been entirely ignored by the software engineering community, as evidenced by ROOM (Real-time Object-Oriented Modeling) [151] and architecture description languages such as Wright [9]. Hardware design languages, such as VHDL, Verilog, and SystemC, are all actor-oriented. In the academic community, active objects and actors [3], I/O automata [126], Polis and Metropolis [35], and Giotto [70], all emphasize actor-orientation.

Domain-specific languages (DSLs) have significant impact on the design process [100]. In embedded systems, where computation and communication interact with the physical world, DSLs offer an effective way to structure information about the system to be designed along the "natural dimensions" of the application [29][36]. We take the position that DSLs for embedded systems should have a mathematically manipulable representation, and call these domain-specific *modeling* languages (DSMLs). The VU team has been researching the use of *meta-languages* to describe DSMLs [105][58]—for example, in work on the Graphical Modeling Environment (GME), they used UML object diagrams and OCL (Object Constraint Language) as a meta-language for modeling abstract syntax. There are many other interesting examples of domain-specific languages targeting multiple-aspect modeling, such as Modelica [135], Rosetta [146], dialects of Real-time UML [44], Charon [13], and others. In this ITR project, we will focus on developing the foundations of model-based design, building on the research conducted to date at UCB and VU. We are explicitly *not* develop yet another representation formalism and design process; rather, we will address the fundamental issues of composition of domain-specific languages, design synthesis using a model-based approach, and transformation of models.

1.2.1 Composition of domain-specific modeling languages

Building the semantic foundations of a DSML and support for an appropriate end-to-end design process is a difficult, expensive and lengthy process. In addition, they are of course domain-specific---and in any complex system, different parts of the system, and different levels in a hierarchy of abstraction, will encompass multiple knowledge domains thus requiring the use of different DSMLs. We intend to resolve these difficulties by developing *language-engineering* support using a meta-modeling approach. By this, we mean that we use a representation formalism (that is, the meta-language) for describing the semantic and syntactic elements of DSMLs. Models built using a DSML, as well as in the meta-language itself, are manipulable, enabling powerful techniques such as deep composition of DSMLs.

The VU team has done work in this area, developing techniques for composing the abstract syntax of DSMLs using meta-model composition [113]. The UCB team has taken a complementary approach, using a uniform abstract syntax in multiple DSMLs, and making the semantic mapping of these DSMLs changeable [39]. They have also performed preliminary research on a type-system for abstract semantic domains [119].

A clear next step is to join the efforts of these two teams. In particular, our goal in this project is to understand the heterogeneous composition of DSMLs using meta-models. Starting with our existing results, we will examine different meta-modeling formalisms and develop composition operators to manipulate meta-models. We will place special emphasis on the composition of meta-models using non-orthogonal component languages. We will also work on the verification of properties of the composed modeling language. Finally, we will investigate how to integrate the representation of the semantic domain and semantic mapping in the meta-modeling framework. This will depend strongly on integrating results from our proposed research in hybrid systems theory with model-based design.

1.2.2 Model synthesis using design patterns

The software design patterns community has developed a number of useful prototypical solutions for re-curring software design problems [55]. Very little is known, however, about design patterns suitable for embedded systems. In previous research, the VU team has developed techniques to precisely formulate design spaces using *templates* and *parameterization*. VU used design constraints and symbolic, design-space pruning methods to synthesize models that meet all requirements [22]. We see the meta-modeling approach to defining the abstract syntax of DSMLs as an opportunity for the introduction and use of de-sign patterns in design synthesis for embedded systems. Graph grammars [147] are a promising candidate for the representation of abstract syntax meta-models—preliminary research at VU [121] has shown that by defining design patterns as parameterized meta-models, they can be used as graph rewriting rules, en-abling the use of graph rewriting technology in the synthesis of new designs. We propose the following:

- Research on pattern-based model synthesis using graph grammars and graph rewriting machinery transparently embedded in domain-specific development environments. This research has many challenges: precise and efficient representation of model transformation and synthesis algorithms as graph rewriting rules; generation of efficient transformation code from the rewriting rules; optimization when the rewriting involves search and non-determinism; integration of the graph-rewriting techniques with traditional (though efficient) procedural approaches; precise semantics of the rewriting; and verification of transformation systems based on graph rewriting.
- Research on the expression of our semantic understanding of models of computation as design constraints and design patterns by integrating VU and UCB work. If successful, this research may have tremendous impact on the productivity of embedded systems design by enabling transformation of designs between different platforms supporting different models of computations.

1.2.3 Model transformation

Model transformation is the workhorse of model-based design. In the successive-refinement model of the design process [149], a design evolves along iterative refinement, fusion, composition, and analysis of models. During this process, design models change not only because of their evolution but also because of the need for transforming domain-specific design models to the native forms of various analysis and synthesis tools. In practice today, many model transformations are performed entirely by hand, essentially by reconstructing the model in a different semantic framework. Our approach will be to use *model-based generators*, which systematize the conversion process. For example, an abstracted dataflow model of the signal processing in an embedded system, annotated with real-time constraints, might be converted to a Giotto model of a multitasking software implementation. Of course, a generator is valid only if well-formed models produce well-formed models. The transformations may add implementation detail (concretization) or remove implementation details (abstraction)—both are useful.

We see our approach as an essential generalization of the two-step process ingrained into traditional software design: detailed design is done in a high-level language, and then compiled to produce executable code. As embedded systems become more complex, however, more and more levels of abstraction appear in the design process. For example, block diagrams in a modeling environment like Simulink might be used to design control systems long before software is written in a high-level language. One can view these executable abstract specifications as designs in "higher-level" languages, and transformations to other high-level languages as new forms of compilation. We will re-examine compiler technology from this point of view, with the goal of replacing the *ad hoc* "code generators" or "auto-coders" that are sometimes used today with a systematic, model-based generation technology.

Meta-generation. Active research by the VU group has focused on meta-generators—generators for generating generators from their specification—in multiple-aspect modeling environments [134]. The approaches investigated included component-based generator composition, and specification of generators using graph rewriting [121]. The UCB group has focused on building generators using a reusable infrastructure [169]. Taking into consideration the tremendous richness of DSMLs and models of computation, we propose a major expansion of this research by combining the VU and UCB results, and developing a new meta-generation technology. We believe that a technology for generator implementation needs to satisfy two key, often contradictory requirements: it is easy to use; and it supports a level of formality such that properties of generators can be determined algorithmically. Techniques based on graph rewriting may not be the only suitable solution, and so we will experiment with other techniques, such as techniques based on algebraic specifications and category theory [155], language-based approaches (PARLANSE from Semantic Designs [152]), and techniques based on functional programming [99] and partial evaluation [102]. Our goal is to create and disseminate a reusable meta-generation framework that can be used by researchers working on embedded systems to rapidly prototype new model transformation tools.

Scalable models. Current model-based design and meta-modeling techniques scale poorly to large systems. For example, the UCB team have worked with staff on the IceCube project at Lawrence Berkeley Labs to produce a complex model for analyzing signals from neutrino detection arrays suspended in the Antarctic ice [123]. A scaled-down model was used, as the visual syntax does not scale well (see the referenced paper for visual syntax examples). Large systems like this can be specified using a model-based generator approach, in which the designer specifies generative components for replicating and conditionally instantiating complex model structures [112]. An IceCube model specified this way would contain a component that defines the structure, and individual detector models as first-class objects provided as arguments to that component [46]. This approach allows models to operate on models, and is related to higher-order functions in functional languages [98][114]; it has also been demonstrated in DSMLs [144][118]. We plan to investigate how this approach can be raised to the meta-level, and how a novel meta-modeling language can allow the definition of such generative modeling languages.

Construction of embeddable generators. To build complex embedded systems that can adapt their architecture at run-time, models need to be deployed on the run-time platform, and the system re-generated when circumstances dictate. This regeneration process necessitates embeddable generators that can re-interpret models and reconfigure the run-time system, under strict time constraints. Starting from preliminary research by the VU group [2], we will investigate the fundamental science underlying the construction of time- and resource-bounded embedded generators, and the technologies required for their implementation.

1.3 Advanced Tool Architectures

We have a long history of producing high-quality pioneering tools (such as Spice, Espresso, MIS, Ptolemy, Polis, and HyTech [74] from UCB, and GME, SSAT, and ACE from UV) to disseminate the results of our research. The conventional notion of "tool," however, does not respond well to the challenges of deep compositionality, rapid construction and composition of DSMLs, and model-based transformation and generation. We therefore propose in this project to shift the emphasis to tool architectures and *tool components*—that is, software modules that can be composed in flexible ways to enable researchers with modest resources to rapidly and (most importantly) correctly construct and experiment with sophisticated environments for hybrid and embedded systems. Concretely, the key products of this work will be a set of toolkits, frameworks [55], and other software modules. We will still develop tools, but only as reference applications of the toolkits and frameworks. This approach has had high impact in other areas, most notably in compilers (for example, [157]), and we believe that its application in the context of MSS will yield high dividends for researchers and tool implementers in hybrid and embedded systems.

The tool architecture work will be guided by the concurrent work in the other research focus areas. In particular, the concept of compositionality of hybrid systems languages and modeling languages has a profound impact on the ways in which we think about tool architectures. We propose to extend the tool architecture work performed to date at UCB and VU. To disseminate the results of this work, we propose to create a national resource consisting of a repository of open-source, well-documented, web-integrated toolkits and frameworks. Particular areas in which we will focus our efforts follow. *Key faculty involved here include Aiken, Henzinger, Karsai, Keutzer, Lee, Messerschmitt, Necula, Sangiovanni-Vincentelli, Sastry, and Sztipanovits.*

1.3.1 Syntax and semantics

The foundation of model-based tools is a suitable abstract syntax. Toolkits operate on this abstract syntax to perform scheduling, resource management, optimization, type inference, type checking, and other functions. Meta-programming frameworks can then be built to implement mappings into the semantic domain.

Semantic composition. If the abstract syntax supports hierarchy, models of computation can be mixed at different levels of the hierarchy. The Ptolemy Project has demonstrated this approach [114] and shown that hybrid systems are a special case of such hierarchical heterogeneity. In Ptolemy, an abstract semantics—which abstracts semantic properties of domain-specific models of computation—enables their hierarchical composition. We will build toolkits and frameworks that support such abstract semantics, and will extend them to non-hierarchical heterogeneous models, such as those where multiple views of a design represent distinct aspects of the design. The frameworks will thus be used to specify semantic interfaces between tools, in contrast to the more commonly addressed problem of syntactic interfaces.

*Visual concrete syntaxes***.** Embedded systems designers are more often application engineers, in fields such as control systems and signal processing, than software engineers. Tools with visual syntaxes make it easier for application engineers to reason about a design and to ensure that it is correct. For researchers to experiment with visual syntaxes, however, is extremely costly. We will develop toolkits and frameworks supporting experimentation with visual syntaxes, new graphical idioms, and modeling languages and language theory that are amenable to rendition in visual syntaxes. These toolkits and frameworks will be based on the meta-programming concepts developed in the GME [113] tool from VU, MetaDome [43] from Honeywell, and Moses [47] from ETH.

*Modal models***.** The figure below is a rendition of a hybrid system rendered as a hierarchical model, where the top level is an automaton, and its middle state is refined to a continuous time model of some physical dynamics. In the continuous-time model, integrators are placed in feedback loops to define second-order differential equations. The pattern that this hierarchical combination follows is that of a *modal model*, in which the states of an automaton represent modes of operation of some other model, and the refinements to the states give the detailed specification of the mode. This pattern can be repeated for concurrency models other than continuous time using exactly the same infrastructure. Toolkits and frameworks supporting modal models, therefore, could be applied not just to hybrid systems, but also to other sorts of modal models. We propose to create such infrastructure.

1.3.2 Interface theories

In the hybrid systems theory focus area, we identified the notion of interface theories as a generalized behavioral type system. For components in embedded systems, this concept is much more powerful than,

for example, the type systems used to capture static structure in object-oriented design. Rather than rely on informal documentation to declare temporal properties, concurrency, and other dynamic properties such as valid ordering of method invocation, we propose to provide infrastructure for interface theories for embedded systems design. Recent work at UCB on interface automata for Ptolemy abstract semantics [119] can be viewed as an implementation of a stateful interface theory for Ptolemy actors. We will continue to develop rigorous abstractions and complementary tool support for interface theories in hybrid models of computation.

1.3.3 Virtual machine architectures

Today, embedded software designers use low-level facilities of a real-time operating system (RTOS), tweaking parameters such as priorities until the system seems to work. The result is, of course, quite brittle. In general-purpose computing, the virtual machine concept has been effective at permitting software developers to focus on problem-level design rather than platform-specific idiosyncrasies. In recent work at UC Berkeley, the virtual machine concept has been applied to embedded systems. The E-machine [77] is a virtual machine that abstracts away from the idiosyncrasies of the RTOS, shifting the burden of ensuring time-correctness from the programmer to the compiler. We will continue to develop platform-independent virtual machines that support the temporal and concurrent properties essential to hybrid and embedded applications. Concepts such as real-time scheduling, exception handling, and code mobility will be handled by the virtual machine rather than being handled in an *ad hoc* way by the application programmer.

1.3.4 Components for embedded systems

One objective of our research is to substantially impact the design of embedded systems in industrial practice. We see reusable components as one means to this end. Although the term "component" is used often in this proposal and elsewhere, there is no broadly accepted definition [30]. Szypersky [161] points out that achieving a component assembly methodology requires a marketplace where a wealth of components is available for purchase and use. Each independent component supplier tries to maximize its market size, justifying the extra time and effort devoted to maximizing the range of feasible uses and simplifying use through appropriate abstraction and configurability and associated tools [37].

Our approach will be to first ask what characteristics of a component would be most beneficial in actual practice in an embedded systems design context. The successful "big component" approach of business software will not work here; instead, embedded software and system design requires a different component technology. A key problem in component systems is providing a coherent framework in which multiple components, likely written in many different programming languages, can interact. Today's component technology does not allow for the tightly-coupled interaction necessary for small components to meet real-time deadlines and other system resource constraints. Part of our work on components, which is aligned with our work on modeling, is to create a framework in which components written in different languages and even with different models of computation can sensibly interact. We will incorporate concepts identified in the work on hybrid systems theory and model-based design into a new component framework suitable for embedded systems. Where necessary, we will assimilate and incorporate ideas from other domains addressing similar challenges, like component software [161], ubiquitous computing (e.g. Jini [159] and PnP [130] and future extensions), web services [56], and the semantic web [25].

1.4 Experimental Research

To be relevant, a new design technology and systems science has to be informed by the real problems of real systems. The investigators in this project have strong track records in developing, fielding, and supporting large-scale applications, including highway transportations systems, air traffic management, electronic design automation systems, aerial robotics, vehicular electronics, sensor nets, and smart structures. Moreover, the participants maintain ongoing collaborations with outside systems efforts, including big science projects (such as the IceCube neutrino detection project), avionics-based homeland defense (the Boeing-led SoftWalls project and a joint NASA-FAA center of excellence), distributed monitoring of critical infrastructure, the Network for Earthquake Engineering Simulation (NEES), and high-performance distributed real-time embedded computing for physics applications (NSF project to build a data acquisitions system for the Fermilab BTEV detector). This ITR project will serve as a focal point for the problems and solutions raised by these systems efforts. A sub-theme here is that the ITR project will offer "new economy" technologies to "old technology" sectors.

Note that we are *not* proposing that the ITR project orbit around a single (or even a small number of) challenge problems or testbeds. The project will engage in dialog with large systems efforts, and provide technology to those systems efforts for evaluation and testing, but it will not divert resources to the large systems building efforts themselves. Our experience indicates that when research is too closely coupled with large challenging applications, delivery of the applications, rather than development of the fundamentals, becomes the primary focus.

Instead, this ITR project will include a distributed laboratory for experimentation with location-aware computing, robotics, aerobotics, vehicle electronics, control systems, security systems, smart environments, smart structures, and even whimsical embedded systems. This laboratory will be used to test the concepts emerging from the center, to evaluate evolving technology from elsewhere, to demonstrate results from the project, and to keep the research grounded in practical, implementable methods. The design of the laboratory, applications, and reference solutions will be made available to the research community. *All ITR investigators will participate in this activity*.

1.4.1 Embedded control systems

This ITR project will evaluate embedded controller design methods for avionics and vehicle electronics by leveraging ongoing efforts at the participating universities. For avionics, we will interact with the teams building software for a set of rotorcraft UAVs at UCB [95], and with the aerospace industry at VU. We will also use an automotive suspension and engine-control testbed at UCB, and interact with the industrial partners of the CHESS center at UCB and the ISIS center at VU. These experimental platforms are being developed with DoD and DARPA funding. Both have potential to address concerns of national security and homeland security needs. These platforms have a substantial amount of embedded real-time software, integrate subsystems that were designed to work independently (for example, sensors from different vendors), and are safety-critical.

At VU, we have been working with Boeing to develop an Embedded Systems Modeling Language that supports the model of computation used in their Bold Stroke architecture for avionics systems. This language allows high-level, component-based modeling of embedded applications in the style of the publish-subscribe paradigm. The models are used not only to synthesize the actual applications, but also to support formal analysis to check schedulability and other properties of the system at design time. VU is also working closely with researchers from the automotive industry to support the component-based design of software for automotive control systems. We have developed techniques for the constraint-based pruning and exploration of complex design spaces consisting of a large number of hardware and software components. The ITR project will allow us to deepen the theoretical foundations of this work.

1.4.2 Embedded software for national and homeland security

UCB is working with Boeing, NASA, and the FAA on avionics approaches to improving homeland defense. One approach being pursued is to create "no-fly zones" that are enforced by the flight control system in aircraft. As an aircraft approaches the boundary of such a zone, the flight control system creates a virtual pushing force that forces the aircraft away. The pilot feels as if the aircraft has hit a soft wall that diverts it.

For fly-by-wire aircraft, this modified control system is conceptually easy to implement. The aircraft carries a three-dimensional model of the earth's atmosphere, annotated with the topology of the surface (which creates real "no-fly zones") and the topology of regulatory constraints on flight space (which creates virtual "no-fly zones"). The virtual no-fly zones would shield, at a minimum, major cities, government and industrial centers, and military installations. The model is updated only rarely, and is coarse grain; the zones are large, representing the overall structure of cities, not individual buildings.

One of the key challenges is that although deployment of the system is conceptually simple in fly-by-wire aircraft, and not that much more difficult in older aircraft, the need for software certification presents a major impediment to deployment. This ITR project will work with Boeing, NASA, and the FAA to begin to identify model-based software design techniques that could lower the certification costs while improving confidence in the software.

1.4.3 Networks of distributed sensors for environmental monitoring and national security

A recent NRC report "Embedded Everywhere" argues that our view of the environment will be transformed by using low-cost wireless distributed sensors. UCB has recently received quite a bit of attention for its wireless "motes" and "smartdust." UCB dropped clusters of these from the air at 29 Palms Marine Corps Station, and monitored electricity consumption in Cory Hall during the height of the power crisis. UCB is currently developing a national experimental test bed on networks of embedded sensors [92] for use by a number of academic and industry partners (with DARPA funding). The protection of critical infrastructures for homeland security can be improved by monitoring them passively using such sensor networks.

Ideally, application scientists should be able to program sensor nets with aggregate query and processing expressions, that would be translated into specific sensor acquisition on the nodes, triggers, interrupts, and communication flow. The problem is related to database query processing, if one views the sensor data as a fine-grained distributed database, with data identified by key, rather than address. Rather than regular tables and records, we have a pool of intermittent, unstructured noisy data streams. One can also view the sensor network as an extremely fine-grained tuple-space, as in Linda or JINI. Many operations are naturally data-parallel but are likely to be statistical, rather than deterministic. We can borrow techniques from online query processing to report the result statistically and with tolerance for adversarial inputs from the nodes. These incremental techniques provide a means of implementing triggers to indicate unusual events. In general, the aggregation operations must handle outliers within the network to be robust to processing and networking errors. We will investigate the right set of data-parallel operations for programming sensor networks.

1.4.4 Hybrid models in structural engineering

Smart Structures are mechanical structures (buildings, aircraft, space systems, maritime vessels, civil structures, and so on) that contain sensors and actuators as integral components of their design and construction. Such configurations, combined with computational capability, enable systems to sense and control the static and dynamic state of the structure. Current applications of such technology include mitigation of extreme stresses, active noise and vibration control, structural health monitoring, structural fault and failure detection, and operational reconfiguration. Smart structures are excellent experimental platforms for networked embedded software applications.

Understanding the behavior of structures under extreme stress is fundamental to engineering robust infrastructure for our society. In near-failure states, behavior of structures is governed by complex interaction of many nonlinear structural components. Complex systems of this nature are best investigated using hybrid simulation techniques. NSF's George E. Brown Network for Earthquake Engineering Simulation (NEES), an $87 million MRI project to be completed in September 2004, will provide a network of structural engineering laboratories, and the hardware foundation for model-based hybrid simulation in structural engineering. This ITR project will provide the science and software foundation for such hybrid simulation.

The methods developed by this ITR will also be evaluated on smart structures in the Vibro-Acoustics Lab at VU. As part of this proposal, we will support the development of three graduated experimental testbeds [52][53][60][156]. The control objective will be to minimize structural vibration caused by exogenous inputs. The testbeds include: a 10-node system acting on a basic, easy-to-model beam experiment; a 30-node beam experiment; and a 50-node rib-stiffened plate experiment. These three experiments will enable investigators to explore the effects of increased complexity both in the structure and in the embedded network. Two important aspects of these testbeds are: (1) that they be accessible and usable by investigators not experienced in applied controls and smart structures technology; and (2) that they relatively inexpensive to construct.

2. Education and Outreach

Our agenda is to build a *modern systems science* (MSS) with profound implications on the nature and scope of computer science and engineering research, the structure of computer science and electrical engineering curricula, and future industrial practice. This new systems science must pervade engineering education throughout the undergraduate and graduate levels. Embedded software and systems represent a major departure from the current, separated structure of computer science (CS), computer engineering (CE), and electrical engineering (EE). In fact, the new, emerging systems science reintegrates information and physical sciences. The impact of this change on teaching is profound, and cannot be confined to graduate level. Based on the ongoing, groundbreaking effort at UCB, we propose to deliberately re-architect and retool undergraduate teaching at the participating institutions, and to make the results widely available to encourage critical discussion and facilitate adoption. In addition, will be recruiting new undergraduate students (usually juniors) from minority institutions through the established REU programs SUPERB-IT at UCB (8 more) and SURGE at VU (5 more) to participate in the research of the project.

2.1 Undergraduate Curriculum Development

Five years ago, UCB EECS faculty initiated a long-term experiment of revising a traditional EE systems science program to better relate to the realities of a computation-oriented world [117]. A traditional EE curriculum, which emphasizes formal modeling and design of physical systems, can be effectively married with a traditional CS curriculum, which emphasizes computation divorced from the physical world. This is not the "standard" CE curriculum, with its focus on the design of computer hardware; rather, it is a new way of thinking about computational systems so that the passage of time, concurrency, resource utilization, and their engagement with the physical world form the basis of system conceptualization, and, inversely, a new way of thinking about physical systems to embrace discrete events and computation. UCB has begun a drastic revision of its undergraduate "EECS" curriculum to introduce a formal systems science to CS undergraduates, and to introduce information science to EE undergraduates. A kingpin of this effort is a sophomore-level course on a computational view of signals and systems [120][170].

UCB and VU will jointly develop more advanced undergraduate courses on system modeling and design, as well as on the underlying hybrid mathematical foundations. We will coordinate discussions on curriculum revision at UCB, VU, and the California community colleges. Based on our experience with new courses and on these discussions, we will seek to revise the undergraduate CS and EE curricula. These curricula are essentially 30 years old, and even a cursory examination reveals the need for reform. For example, concurrency is treated in the ACM CS curriculum mainly in the operating systems course, in the form of mutual exclusion protocols. From the perspective of embedded software and systems this is a misplaced and insufficient way of teaching concurrency, and its focus is inappropriate: concurrency in modern computing systems is not a reducible extension of sequential behavior, but arguably the most fundamental organizing principle of design. The current treatment of real time, with its preoccupation on scheduling, is similarly outdated. A curriculum based on modern systems science will elevate topics such as concurrency and real time from a discussion of implementation details on von Neumann platforms to intrinsic properties of systems and their mathematical models. A systematic infusion of MSS into the undergraduate curriculum has far-reaching consequences. We should not train engineers, as we do today, to become experts in the combinatorial complexities arising in discrete mathematics (CS), or experts in the analytic complexities arising in continuous mathematics (EE). We need engineers who are equally at home in both worlds.

2.2 Undergraduate Curriculum Transfer

The insertion of new course material and laboratory practice will be implemented in the framework of a carefully planned, gradual, measured, and controlled process. The gradual introduction is a key requirement. Over its 30-year history, the undergraduate CS and EE curricula have acquired an interlocking structure of prerequisites, so that changes introduced in one course require accommodating

changes in other courses. We envision a process that will gain acceptance and momentum over the life-span of the project, starting with existing curricula at our institutions, and leading to the implantation of major elements of the MSS agenda. We propose the following steps for the transfer:

1. Develop the goals of new CS and EE curricula based on MSS. The high-level goals will be developed and documented according to ABET requirements, and they will reflect what is to be achieved by the end of the project. We will also develop course evaluation criteria, which we will use later to measure the success of the new courses. **Timeline**: months 1-6 of the project.
2. Develop new "EECS" courses, and corresponding courseware material, that will form the center of the new CS and EE curricula. We will (1) identify what courses are necessary to achieve the curriculum goals, (2) classify these courses into core and elective categories, and (3) study how the introduction of these new courses affects the existing CS and EE courses. **Timeline**: from month 6 to the end of the project.
3. Introduce new core courses into the current CS and EE curricula. Each new core course must replace an existing course requirement. However, during a transition period, the new courses may co-exist with more specialized courses from the "old" curricula, and emphasize the integrated nature of the material. We expect the transition to be uneven—some institutions will respond more quickly to some courses than others. A successful example of the introduction of a new core course is EECS 20 at UCB [120][170]. A second new core course may focus on the mathematical foundations of MSS. We will insist that the faculty from all participating institutions share courseware development—texts, web-based material, laboratory experiments, exercises, and challenge problems. This will ensure accommodation of the diversity among us in terms of student background and faculty orientation and objectives, and aid in the wider adoption of these courses. **Timeline**: years 1-2.
4. Evaluate and revise the new core courses as needed, and adjust (or phase out) the existing CS and EE courses on related (and overlapping) material accordingly. **Timeline**: years 1-2.
5. Introduce new elective courses that specialize in various aspects of embedded and hybrid systems design. **Timeline**: years 3-4.
6. Evaluate and revise the elective courses as needed, and fine-tune the overall curriculum.

2.3 Summer Internship Program in Hybrid and Embedded Software Research (SIPHER)

One of our major challenges in CS and EE education is to attract talent in research careers. America's prosperity in the new millennium and increasing national security concerns make it vitally important to increase the number of students who are interested in joining the nation's technical enterprise as researchers. The situation is particularly serious in the area of women and minorities. While in all areas in our economy we must increasingly build on the diversity within this country, the trends in CS and EE research go in the opposite direction. The potential impact of this trend is extremely negative. On one hand, it contributes to the chronic lack of US graduate students this area, but even more importantly, it makes it very hard to recruit faculty from underrepresented groups, so the trend may perpetuate itself.

The SIPHER program will focus on this problem. We will design the program to motivate the best undergraduate students from women and minority groups to go to graduate programs in CS and EE. The technical area of embedded systems is exceptionally suitable for this. The physicality of embedded systems has a tremendous appeal for students because of the opportunity for active experimentation and interaction with the created "smart objects."

The Director of the SIPHER program, Dr. Brian N. Williams (who is the Assistant Dean for Student Affairs at VU School of Engineering), will pursue an outreach effort toward HBCUs, traditionally white institutions (TWIs) with a critical mass of minority students, and Community Colleges in the South and in California. Taking advantage of Dr. Williams' current responsibility at VU as an Assistant Dean and previous experience managing and directing a summer research program at the University of Georgia (which brought in minority high school students and undergraduates from public HBCUs in the state of Georgia with an interest in science and math), he will establish recruiting networks and linkages to schools such as Fisk University, Tennessee State University, University of Alabama-Birmingham, Morehouse College, Spelman College, Florida State University, Florida A & M University, Tuskegee University, and others. In the State of California, the focus of the recruiting effort will be the California community colleges with which UCB has established relationships and which have a high enrollment of Hispanic and African American students, a number of whom transfer to Berkeley in their junior year.

By utilizing NSF funding and cost sharing from UCB and VU, the SIPHER program will offer summer internship opportunities that partner the best students and faculty from UCB and VU with 10-12 students and 3-5 faculty members from our pool of partnering institutions. To further enhance faculty support from partnering institutions, we propose covering 50% of their salary for 3 summer months. For participating faculty, we will organize summer schools on how to teach the new, required sophomore-level course on signals and systems, as well as other new core courses. Interaction with instructors will also affect the material that we include in the new core courses. Introducing changes in the undergraduate curriculum is difficult. Achieving the far-reaching shifts envisaged by the modern systems science perspective is an even greater challenge. We are committed to bringing this to fruition in our own institutions, and we will strive to facilitate change in other universities and colleges whose faculty wish to emulate our example.

4. Institutional Participation, Leadership, Management, and Organization

4.1 Institutional Participation

This project incorporates the strengths of researchers from 3 institutions, using their expertise and experience to bring the project vision to reality. The project team includes two major research groups with well-established, complementary research programs in hybrid and embedded software. The Berkeley (UCB) group has a long history and leading role in the development of hybrid systems theory (Robotics and Intelligent Systems Laboratory, Partners for Automation on the Highways, Embedded Systems Laboratory) and advanced tool architectures for embedded systems (Ptolemy and Polis groups). The Vanderbilt (VU) group has over a decade history and runs a major research program in model-integrated computing, a foundation of model-based design (Institute for Software Integrated Systems). In the last three years, Berkeley and Vanderbilt have started cooperating in several research programs, leading to the recognition of the benefits of joining their unique perspectives in hybrid and embedded software systems. A leading researcher from the University of Memphis, working on the mathematical foundations of phase transitions in constrained systems, will complement the UCB and VU groups.

4.2 Leadership and Management

The organization chart for the project management is shown in the figure below. The Project Director, Prof. Sastry, will be responsible for the overall management of the project and will chair the Executive Council. The role of the Executive Council is to perform strategic planning, research and outreach coordination, budget allocation, and measurement of the effectiveness of the overall research, education, and dissemination efforts. The members of the Executive Council are Profs. Henzinger, Lee and Sangiovanni-Vincentelli (UCB), and Prof. Sztipanovits (VU). Besides their academic credentials, members of the Executive Council have substantial research management experience. Prof. Sastry is the Chair of the EECS Department at Berkeley, and he was former director of DARPA-ITO. Prof. Sztipanovits is founding director of ISIS, and was a program manager and the deputy director of DARPAITO.
Profs Henzinger, Lee, and Sangiovanni-Vincentelli have been managing major research labs at UCB, and have had extensive consulting and management responsibilities in start-ups and established EDA companies.

The members of the Executive Council are the principal investigators of the four major research focus areas: Prof. Henzinger – Hybrid Systems Theory; Prof. Sztipanovits – Model-based Design, Prof. Lee – Advanced Tool Architectures; and Prof. Sangiovanni-Vincentelli – Experimental Research. The research focus areas will be complemented by the Curriculum Development and Education Outreach areas. The leader of Curriculum Development is Prof. Varaiya (UCB); the leader of Education Outreach is Prof. Karsai (VU), in cooperation with Dr. Brian Williams (VU). Each leader will be responsible for developing the research and technology activities of the research focus area. Each area will have a number of projects; the investigators in these projects constitute the focus area committee and are

responsible for executing the focus area research agenda.

Director
Shankar S. Sastry
Executive Council
Shankar S. Sastry
Thomas A. Henzinger
Edward A. Lee
Alberto Sangiovanni-Vincentelli
Janos Sztipanovits
Hybrid Systems Theory
Thomas A. Henzinger
Advanced Tool Architectures
Edward A. Lee
Model-based Design
Janos Sztipanovits
Experimental Research
Alberto Sangiovanni-Vincentelli
Board of Directors
Curriculum Development
Pravin Varaiya
Education Outreach
Gabor Karsai

The project team will be supported by a Board of Advisors chosen from industry and academia. The primary role of the Board of Advisors is to provide feedback to the team on progress and effectiveness in industrial applications. Members will be major industry stakeholders in advances in embedded software technology (GM, FORD, Boeing, Honeywell, Lockheed-Martin, Motorola, Intel), representatives from the tool industry (Cadence, Synopsys), and representatives from other academic groups such as Stanford, the University of Pennsylvania, the DARPA-SRC MARCO Research Centers (including CMU and Georgia Tech), the Technical University of Vienna, and the EU Group on Embedded Software, Artiste, headed by Prof. Sifakis from CNRS-VERIMAG, Grenoble. Letters of support from a representative sample of Board Members is attached to this proposal. The Board of Advisors will meet twice per year to review research activities. The semi-annual meetings will alternate between California and Tennessee and will have a retreat format with participating faculty, students, and visiting researchers presenting research results, progress, and future directions. The project team will meet immediately after each of the retreats, to collect the feedback from the retreats and determine new ideas and directions with an emphasis on the individual focus areas. In addition, the investigators will meet in conjunction with the annual Embedded Software Workshop, EMSOFT, established by Henzinger and Kirsch in 2001, and co-chaired by Sangiovanni-Vincentelli in 2002. The theme of this meeting will be integration among research focus areas and curriculum/education outreach. Coordinating this meeting with EMSOFT will enable us to involve a broader group of researchers from the community.
In addition, teleconferences and net meetings with colleagues will be used on an as-needed basis for consultation and planning. The Executive Council will have a formal teleconference arrangement each month for operational issues.

4.3 Institutional Commitment

The administrative units for the research effort at the two lead campuses are the Center for Hybrid and Embedded Software (CHESS) at UCB and the Institute for Software Integrated Systems (ISIS) at VU. This proposal to the NSF-ITR will serve as an incubator to the UCB Center on Hybrid and Embedded Software Systems (CHESS). UCB-CHESS will be a research unit operating as a part of the newly established initiative at the University of California, CITRIS, the Center for Information Technology in the Interests of Society (see the letter of support from Dean Richard Newton). CITRIS Director Prof. Bajcsy is an investigator on this proposal. They will provide a point of greater outreach of the research of the center to societal-scale information systems, and will also provide grant management and personnel support. CITRIS involves a commitment by the State of California of $100M over the period 2001–2004. The majority of the state funds will be used for the construction of new laboratories in Cory Hall and a new building located next door to Cory Hall and Soda Hall (where all UCB investigators reside), as well as networking, storage, and other infrastructure needs. The laboratories for the experimental components of CHESS will be housed in CITRIS space. CITRIS is organized according to a center of centers concept and is designed to address societal-scale concerns and applications-pull parts of the IT research agenda. CHESS will primarily do technology development with a view to integration in embedded platforms in IT and old economy industries. CHESS Center organization will consist of a Scientific Director (Prof. Henzinger), Participating Faculty, an Executive Director (to be hired), and a Board of Advisors. The Scientific Director and the Participating Faculty will direct all research activities. The Executive Director will be responsible for Center operations and business management, including industrial sponsorship.
The VU Institute for Software Integrated Systems (ISIS) focuses on conducting and disseminating research on model-integrated computing and its applications. ISIS includes academic faculty, graduate students, professional research staff, and a wide range of industrial collaborators. ISIS is supported by the Model-Integrated Computing Alliance, whose members are private industry and government labs. The ISIS director, Prof. Sztipanovits, will be responsible for integrating and coordinating the proposed research and teaching activities of ISIS researchers and the participating research teams from VU and MU. Beside his academic credentials, Prof. Sztipanovits brings a great deal of program management experience with industry (especially the automotive and semiconductor industry) and DARPA. ISIS has extensive support from the Vanderbilt University School of Engineering (VUSE). VUSE provides over $500,000 annual operating budget for the ISIS director. The School will also provide funding for the renovation of a 13,000 ft^2 building in 2002. The building will have sufficient space to run the proposed outreach program and host the increased number of visiting researchers, faculty, and students. Besides commitment from the School, the VU EECS department strongly supports the major curriculum modernization activity of the proposed effort.

4.4 Leadership in the Field and Outreach to Other Groups

The primary focus of our ITR submission is to develop the engineering science and methods for the modern systems science of hybrid and embedded systems. In addition to this fundamental rethinking, the tools and technologies that we will develop will impact systems for vehicle electronics, avionics, networked embedded sensors, and other safety-critical systems. CHESS at UCB and ISIS at VU will have an outreach effort to industry in the areas of avionics, vehicle electronics, networked embedded systems, and high-confidence systems. We expect to engage industry leaders from these market sectors in the Advisory Board (see the letters of support from Boeing, Ford, GM, and Lockheed Martin), and will eventually rely on industry to dedicate researchers and case studies to the ITR Project. Case studies are critical, since they will serve as calibration points of the level of maturity of the design tools and also enable us to confront two often-quoted difficulties in industry: (1) software advances lag behind hardware advances; and (2) there are numerous cost overruns in the delivery of embedded software. This latter difficulty results in the "testing until the money runs out" mindset, which in turn is reflected as cost overruns.

This proposal has an ambitious research agenda, which will establish a well-structured new core for embedded software including a new systems science, model-based design technology, tool architectures, experimental platforms, and education. We are fully aware of the importance of metrics that will be used to measure the success of the proposed ITR. While we have an overarching vision of the direction of the project, its instantiation can be and will be measured by the number of special conferences, special sessions at conferences, journals, and intellectual excitement in the scientific community. A number of activities and opportunities, listed below, have allowed us to communicate our vision nationally and internationally (see the letters of support from the European Network on Embedded Software Artiste and the letter from the Technical University of Vienna). We will continue to pursue the following and other such activities in the course of this project.

 Hybrid Systems---Control and Computation. An annual workshop initiated by two of the principal investigators, Henzinger and Sastry, and co-chaired by Sangiovanni-Vincentelli in 2001.
 International Workshop on Embedded Software. An annual workshop, in which Henzinger, Lee, and Sztipanovits have played a key role and co-chaired by Sangiovanni-Vincentelli in 2002.
 New Visions in Software Design and Productivity (SDP). A workshop co-chaired by Sztipanovits, and funded the Interagency Working Group for Information Technology Research and Development (ITRD)
 IEEE Proceedings, Special Issue on Embedded Software Design. Sztipanovits and Sastry are coeditors of an IEEE Proceedings Special Issue to appear in early 2003.
 National Experimental Platform for Hybrid and Embedded System Technology (NEPHEST). UCB and VU are participants in a seedling effort coordinated by Lockheed Martin on the design of tools and frameworks for embedded software design.
 Model Integrated Computing Alliance. ISIS established the Model-Integrated Computing Alliance in 1995. The members are Boeing, DuPont, Motorola, GM-Saturn, IBM, USAF Arnold Center, NASA Marshall Flight Sciences Center, and Sandia National Labs.

References Cited

[1] M. Abadi and L. Lamport, "Conjoining Specifications," *ACM Transactions on Programming Languages and Systems*, Vol 17, pp 507–534, 1995.

[2] B. Abbott, T. Bapty, C. Biegl, G. Karsai, and J. Sztipanovits, "Model-Based Approach for Software Synthesis," *IEEE Software*, pp. 42-53, May, 1993.

[3] G. A. Agha, Actors: A Model of Concurrent Computation in Distributed Systems, MIT Press, Cambridge, MA, 1986.

[4] A. Aiken, M. Fähndrich, J. Foster, and Z. Su, "A Toolkit for Constructing Type- and Constraint-Based Program Analyses" (invited paper).,In *Proceedings of the 2nd International Workshop on Types in Compilation, Kyoto, Japan, LNCS #1473, pages 76-96, March 1998*

[5] A. Aiken, M. Fähndrich, and Z. Su, "Detecting Races in Relay Ladder Logic Programs," In *Proceedings of the 1st International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Lisbon, Portugal, pages 184-200, April, 1998.

[6] L. de Alfaro and T.A. Henzinger, "Interface automata," Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering, ACM Press, 2001.

[7] L. de Alfaro and T.A. Henzinger, "Interface theories for component-based design," *EMSOFT 01: Embedded Software*, eds T.A. Henzinger and C.M. Kirsch, Lecture Notes in Computer Science 2211, Springer-Verlag, 2001, pp. 148–165.

[8] L. de Alfaro and T.A. Henzinger and R. Jhala, "Compositional methods for probabilistic systems," *CONCUR 01: Concurrency Theory*, Lecture Notes in Computer Science 2154, Springer-Verlag, 2001, pp. 351–365.

[9] R. Allen and D. Garlan, "Formalizing Architectural Connection," in *Proc. of the 16th International Conference on Software Engineering* (ICSE 94), May 1994, pp. 71-80, IEEE Computer Society Press.

[10] R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G. Pappas, H. Rubin, and J. Schug. "Hybrid modeling and simulation of biological systems," *Fourth International Workshop on Hybrid Systems: Computation and Control*, LNCS 2034, pp. 19--32, 2001.

[11] R. Alur, C. Courcoubetis, and T.A. Henzinger. "Computing accumulated delays in real-time systems." *Formal Methods in System Design*, 11:137–156, 1997.

[12] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P. H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. "The algorithmic analysis of hybrid systems." *Theoretical Computer Science*, 138:3–34, 1995.

[13] R. Alur, T. Dang, J. Esposito, R. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G. Pappas, and O. Sokolsky, "Hierarchical Hybrid Modeling of Embedded Systems." *Proceedings of EMSOFT'01: First Workshop on Embedded Software*, October 8-10, 2001.

[14] R. Alur and D.L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, Vol 126, pp 183–235, 1994.

[15] R. Alur, T. Feder, and T.A. Henzinger. "The benefits of relaxing punctuality." *Journal of the ACM*, 43:116–146, 1996.

[16] R. Alur, R. Grosu, Y. Hur, V. Kumar and I. Lee, "Modular specification of hybrid systems in Charon," in *Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, Springer-Verlag, 2000, pp 130–144.

[17] R. Alur and T.A. Henzinger. "Modularity for timed and hybrid systems." In A. Mazurkiewicz and J. Winkowski, editors, *CONCUR 97: Concurrency Theory*, Lecture Notes in Computer Science 1243, pages 74–88. Springer-Verlag, 1997.

[18] R. Alur and T.A. Henzinger. "Real-time system = discrete system + clock variables." *Software Tools for Technology Transfer*, 1:86–109, 1997.

[19] R. Alur and T.A. Henzinger. “Reactive modules.”*Formal Methods in System Design*, 15:7–48, 1999.

[20] R. Alur, T.A. Henzinger, and P. H. Ho. “Automatic symbolic verification of embedded systems.” *IEEE Transactions on Software Engineering*, 22:181–201, 1996.

[21] R. Alur, T.A. Henzinger, and H. Wong-Toi. “Sy mbolic analysis of hybrid systems.” In *Proceedings of the 36th International Conference on Decision and Control*, pages 702–707. IEEE Press, 1997.

[22] T. Bapty, S. Neema, J. Scott, J. Sztipanovits, and S. Asaad, “Model-Integrated Tools for the Design of Dynamically Reconfigurable Systems,” *VLSI Design*, 10, 3, pp. 281-306, 2000.

[23] M. di Benedetto and A. Sangiovanni-Vincentelli (eds), *Proceedings of the Fourth International Workshop on Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2034, Springer-Verlag, 2001.

[24] A. Benveniste and G. Berry, “The Synchronous Approach to Reactive and Real-Time Systems,” *Proceedings of the IEEE*, vol. 79, no. 9, 1991, pp. 1270-1282.

[25] T. Berners-Lee, J. Hendler, and O. Lassila. "The Semantic Web". *Scientific American*, May 2001. Available at http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html, March 8, 2002.

[26] S. S. Bhattacharyya, P. K. Murthy, and E. A. Lee, *Software Synthesis from Dataflow Graphs*, Kluwer Academic Press, 1996.

[27] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson, (2001), “The scaling window of 2-SAT transition,” *Random Structures and Algorithms* 18, 201–256.

[28] R. H. Bourgonjon, “Embedded Systems in Consumer Products,” in *Lecture Notes on Embedded Systems*, LNCS Vol. 1494, 1996, pp. 395-403.

[29] F. P. Brooks, Jr., “No Silver Bullet: Essence and Accidents of Software Engineering”, *Computer Magazine*; April 1987.

[30] M. Broy, A. Deimel, J. Henn, K. Koskimies, F. Plasil, G. Pomberger, W. Pree, M. Stal, and C. Szyperski, “What characterizes a (software) component?”. *Software-Concepts and Tools*, vol.19, (no.1), 1998. p.49-59.

[31] M. C. Çavuşoğlu, M. Cohn, F. Tendick and S. S. Sastry, “Laparoscopic Telesurgical Workstation,” *Proceedings of the SPIE International Symposium on Biological Optics (BIOS '98)*, January 1998.

[32] M. C. Çavuşoğlu and F. Tendick and M. Cohn and S. S. Sastry, “A Laparoscopic Telesurgical Workstation,” *IEEE Transactions on Robotics and Automation*, Vol 15, Nr 4, August 1999.

[33] M. C. Çavuşoğlu and J. Yan and S.~S. Sastry, “A Hybrid System Approach to Contact Stability and Force Control in Robotic Manipulators,” pp 143–148, *Proceedings of the 12th IEEE International Symposium on Intelligent Control (ISIC'97)*, 1997.

[34] M. C. Çavuşoğlu, *Control of a Telesurgical Workstation*, University of California at Berkeley, Electronics Research Laboratory Memo M97/3, May 1997. Also M.S. Project Report, University of California at Berkeley, May 1997.

[35] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, A. Sangiovanni-Vincentelli, “A Formal Methodology for Hardware/Software Co-design of Embedded Systems,” *IEEE Micro*, August 1994, pp.26-36. and F. Balarin, L Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, Y. Watanabe, G. Yang, Concurrent Execution Semantics and Sequential Simulation Algorithms for the Metropolis Meta-Model, *Proc. CODES 2002.*

[36] T. Clark, A. Evans, S. Kent, and P. Sammut: “The MMF Approach to Engineering Object-Oriented Design Languages,” *Workshop on Language Descriptions, Tools and Applications (LDTA2001)*, April, 2001

[37] Cox, B. *Superdistribution: Objects as Property on the Electronic Frontier*; Addison-Wesley 1996. Available at http://www.virtualschool.edu/mon, March 8, 2002.

[38] Crawford, J. M. and Auton, L. D. (1993), *Proc. 11th Natl. Conf. on Artificial Intelligence*, AAAI Press, Menlo Park, CA, 21.

[39] J. Davis II, C. Hylands, B. Kienhuis, E. A. Lee, J. Liu, X. Liu, L. Muliadi, S. Neuendorffer, J. Tsay, B. Vogel, and Y. Xiong, “Heterogeneous Concurrent Modeling and Design in Java,” *Technical Memorandum UCB/ERL M01/12*, EECS, University of California, Berkeley, March 15, 2001. http://ptolemy.eecs.berkeley.edu/publications.

[40] R. DeLine and M. Fähndrich. “Enforcing high-level protocols in low-level software.” In *Proceedings of the ACM Conference on Programming Language Design and Implementation*, 2001, pages 59-69.

[41] J. Desharnais, V. Gupta, R. Jagadeesan and P. Panangaden. “Metrics for Labeled Markov Systems,” In *Proceedings of the Tenth International Conference on Concurrency Theory (CONCUR)*, Lecture Notes in Computer Science 1664, Springer-Verlag, 1999.

[42] D.L. Dill, Trace Theory for Automatic Hierarchical Verification of Speed-independent Circuits, MIT Press, 1989.

[43] Dome Official Web Site, at http://www.htc.honeywell.com/dome/.

[44] B. P. Douglass, Real-Time UML: Efficient Objects for Embedded Systems. Addison-Wesley-Longman, 1997.

[45] M. S. Downes and M. C. Çavuşoğlu and W. Gantert and L. W. Way and F. Tendick, “Virtual Environments for Training Critical Skills inLaparoscopic Surgery,” pp 316–322, *Proceedings of Medicine Meets Virtual Reality VI (MMVR '98)*, January 1998.

[46] R. Esser and J. W. Janneck, “Exploratory Performance Evaluation using Dynamic and Parametric Petri Nets,” *Proc. High Performance Computing*, 2000.

[47] R. Esser and J. W. Janneck, “Moses: A tool suite for visual modeling of discrete-event systems,” in *Symposia on Human-Centric Computing (HCC ‘01)*, pp 272–279, September 2001.

[48] D. Estrin (ed), *Embedded Everywhere*, NRC Study, 2001.

[49] R. S. Fearing and G. Moy and E. Tan, “Some Basic Issues in Teletaction,” Vol 4, pp 3093–3099, *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997.

[50] J. Foster, T. Terauchi, and A. Aiken, “Flow-Sensitive Type Qualifiers,” *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, to appear, June 2002.

[51] J. Foster, M. Fähndrich, and A. Aiken, “A Theory of Type Qualifiers,” *Proceedings of the SIGPLAN Conference on Programming Language Design and Implementation*, pages 192-203, Atlanta, Georgia, June 1999.

[52] K. D. Frampton and R. L. Clark, “Control of Sound Transmission through a Convected Fluid Loaded Plate with Piezoelectric Sensoriactuators,” *Journal of Intelligent Materials Systems and Structures*, Vol. 8, No. 8, pp. 686-696, August 1997.

[53] K. D. Frampton, R. L. Clark and E. H. Dowell, “Active Control of Panel Flutter with Linearized Potential Flow Aerodynamics,” *Journal of Aircraft*, Vol. 33, No. 4, pp. 768-774, July-August 1996.

[54] A. Frieze and S. Suen, (1996), “Analysis of two simple heuristics for a random instance of *K*-SAT,” *Journal of Algorithms* 20, 312–335.

[55] E. Gamma, and R. Helm, and R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison Wesley, 1994.

[56] T. Gardner, “An introduction to Web services”, *Ariadne*, (no.29), UKOLN, 3 Oct. 2001.

[57] A. Girault, B. Lee, and E. A. Lee, “Hierarchical Finite State Machines with Multiple Concurrency Models,” *IEEE Transactions On Computer-aided Design Of Integrated Circuits And Systems*, Vol. 18, No. 6, June 1999.

[58] J. Gray, T. Bapty, and S. Neema, “Handling Crosscutting Constraints in Domain-Specific Modeling,” *Communications of the ACM*, October, 2001.

[59] M. Greenstreet and I. Mitchell, "Reachability Analysis Using Polygonal Projections," *Proc. 2nd International Workshop on Hybrid Systems: Computation and Control*, pp 103-116, March 1999.

[60] C. Guigou, C. R. Fuller and K. D. Frampton, "Experiments on Active Control of Acoustic Radiation due to a Clamped Edge on a Semi-Infinite Beam," *Journal of Sound and Vibration*, Vol. 169, No. 4, pp. 503-526, January 1994.

[61] V. Gupta, T.A. Henzinger, and R. Jagadeesan. "Robust timed automata." In O. Maler, editor, *HART 97: Hybrid and Real-time Systems*, Lecture Notes in Computer Science 1201, pages 331–345. Springer-Verlag, 1997.

[62] N. Halbwachs, *Synchronous Programming of Reactive Systems*, Kluwer Academic Publishers, 1993.

[63] D. Harel, "Statecharts: A Visual Formalism for Complex Systems," *Sci. Comput. Program*., vol 8, pp. 231-274, 1987.

[64] T.A. Henzinger, "Masaccio: a formal model for embedded components," *TCS 00: Theoretical Computer Science*, eds. J. van Leeuwen and O. Watanabe and M. Hagiya and P.D. Mosses and T. Ito, Lecture Notes in Computer Science 1872, Springer-Verlag, 2000, pp. 549–563.

[65] T.A. Henzinger. "The theory of hybrid automata." In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.

[66] T.A. Henzinger. "It's about time: Real-time logics reviewed." In D. Sangiorgi and R. de Simone, editors, *CONCUR 98: Concurrency Theory*, Lecture Notes in Computer Science 1466, pages 439–454. Springer-Verlag, 1998.

[67] T.A. Henzinger. "Masaccio: A formal model for embedded components." In T. Ito, editor, *Proceedings of the First IFIP International Conference on Theoretical Computer Science*, Lecture Notes in Computer Science 1872, pages 549–563. Springer-Verlag, 2000.

[68] M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. "Computing simulations on finite and infinite graphs." In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 453–462. IEEE Computer Society Press, 1995.

[69] T.A. Henzinger, B. Horowitz, and C.M. Kirsch. "Embedded control systems development with Giotto." In *Proceedings of the SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, ACM Press, 2001.

[70] T. A. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A Time-Triggered Language for Embedded Programming," *Proc. of EMSOFT 2001*, pp 166—184, Tahoe City, CA, *LNCS 2211,* Springer-Verlag, *October*, 2001.

[71] T.A. Henzinger, B. Horowitz, and R. Majumdar. "Rectangular hybrid games." In J.C.M. Baeten and S. Mauw, editors, *CONCUR 99: Concurrency Theory*, Lecture Notes in Computer Science 1664, pages 320–335. Springer-Verlag, 1999.

[72] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. "Beyond HyTech: Hybrid systems analysis using interval numerical methods." In N.A. Lynch and B. Krogh, editors, *HSCC 00: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, pages 130–144. Springer-Verlag, 2000.

[73] T.A. Henzinger, P. H. Ho, and H. Wong-Toi. "HyTech: The next generation." In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 56–65. IEEE Computer Society Press, 1995.

[74] T.A. Henzinger, P. H. Ho, and H. Wong-Toi. "HyTech: A model checker for hybrid systems." In O. Grumberg, editor, *CAV 97: Computer-aided Verification*, Lecture Notes in Computer Science 1254, pages 460–463. Springer-Verlag, 1997.

[75] T.A. Henzinger, P. H. Ho, and H. Wong-Toi. "HyTech: A model checker for hybrid systems." *Software Tools for Technology Transfer*, 1:110–122, 1997.

[76] T.A. Henzinger, P. H. Ho, and H. Wong-Toi. "Algorithmic analysis of nonlinear hybrid systems." *IEEE Transactions on Automatic Control*, 43:540–554, 1998.

[77] T. A. Henzinger and C. M Kirsch, "The Embedded Machine: Predictable, Portable Real-Time Code," *Proc of Programming Language Design and Implementation*, June 2002 (to appear).

[78] T.A. Henzinger and P.W. Kopke. "State equivalences for rectangular hybrid automata." In U. Montanari and V. Sassone, editors, *CONCUR 96: Concurrency Theory*, Lecture Notes in Computer Science 1119, pages 530–545. Springer-Verlag, 1996.

[79] T.A. Henzinger and P.W. Kopke. "Discrete-time control for rectangular hybrid automata." In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *ICALP 97: Automata, Languages, and Programming*, Lecture Notes in Computer Science 1256, pages 582–593. Springer-Verlag, 1997.

[80] T.A. Henzinger and P.W. Kopke. "Discrete-time control for rectangular hybrid automata." *Theoretical Computer Science*, 221:369–392, 1999.

[81] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. "What's decidable about hybrid automata?," *Journal of Computer and System Sciences*, 57:94–124, 1998.

[82] T.A. Henzinger and O. Kupferman. "From quantity to quality." In *HART 97: Hybrid and Real-time Systems*, Lecture Notes in Computer Science 1201, pages 48–62. Springer-Verlag, 1997.

[83] T.A. Henzinger, O. Kupferman, and M.Y. Vardi. "A space-efficient on-the-fly algorithm for real-time model checking." In *CONCUR 96: Concurrency Theory*, Lecture Notes in Computer Science 1119, pages 514–529. Springer-Verlag, 1996.

[84] T.A. Henzinger and R. Majumdar. "Symbolic model checking for rectangular hybrid systems." In S. Graf and M. Schwartzbach, editors, *TACAS 00: Tools and Algorithms for the Construction and Analysis of Systems*, Lecture Notes in Computer Science 1785, pages 142–156. Springer-Verlag, 2000.

[85] T.A. Henzinger, M. Minea, and V. Prabhu. "Assume-guarantee reasoning for hierarchical hybrid systems." In *HSCC 01: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 2034, pages 275–290. Springer-Verlag, 2001,

[86] T.A. Henzinger and V. Rusu. "Reachability verification for hybrid automata." In T.A. Henzinger and S. Sastry, editors, *HSCC 98: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1386, pages 190–204. Springer-Verlag, 1998.

[87] T.A. Henzinger and J. F. Raskin. "Robust undecidability of timed and hybrid systems." In N.A. Lynch and B. Krogh, editors, *HSCC 00: Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, pages 145–159. Springer-Verlag, 2000.

[88] T.A. Henzinger, J. F. Raskin, and P. Y. Schobbens. "The regular real-time languages." In K.G. Larsen, S. Skyum, and G. Winskel, editors, *ICALP 97: Automata, Languages, and Programming*, Lecture Notes in Computer Science 1443, pages 580–591. Springer-Verlag, 1998.

[89] T.A. Henzinger and S. Sastry (eds), *Proceedings of the First International Workshop on Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1386, Springer-Verlag, 1998.

[90] T.A. Henzinger and H. Wong-Toi. "Linear phase-portrait approximations for nonlinear hybrid systems." In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 377–388. Springer-Verlag, 1996.

[91] T.A. Henzinger and H. Wong-Toi. "Using HyTech to synthesize control parameters for a steam boiler." In J. R. Abrial, E. Borger, and H. Langmaack, editors, *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, Lecture Notes in Computer Science 1165, pages 265–282. Springer-Verlag, 1996.

[92] J. Hill, R. Szewczyk, A. Woo, D. Culler, S. Hollar, K. Pister, "System Architecture Directions for Networked Sensors," *9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov. 2000, pp. 93-104

[93] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice-Hall, 1985.

[94] K. H. Hodges, Embedded Micro-component Market Study, DARPA.

[95] B. Horowitz, J. Liebmann, C. Ma, A. Sangiovanni-Vincentelli, and S. Sastry "Platform Based Software Design and Systems Integration for Unmanned Aerial Vehicles," submitted to the *Proceedings of the IEEE, Special issue in Embedded Software*, 2003.

[96] L. Howard, *Courseware Authoring Technology project*. Online at http://www.isis.vanderbilt.edu/projects/VaNTH/index.htm.

[97] J. Hu, J. Lygeros and S. Sastry, "Towards a theory of Stochastic Hybrid Systems", in *Proceedings of the 3rd Workshop on Hybrid Systems: Computation and Control, Pittsburgh*, 2000.

[98] P. Hudak, "Conception, Evolution, and Application of Functional Programming Languages," *ACM Computing Surveys*, Vol. 21, No. 3, September 1989.

[99] P. Hudak, The Haskell School of Expression: Learning Functional Programming through Multimedia, Cambridge University Press,New York, 2000.

[100] P. Hudak: *Keynote address at the Usenix DSL Conference*, 1997, http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/hudak-paul/hudak-dir/dsl/index.htm.

[101] R. E. Johnson, "Frameworks = (Components + Patterns)," *Communications of the ACM*, Vol. 40, No. 10, pp. 39-42, October 1997.

[102] N. Jones, C. Gomard, and P. Sestoft, *Partial Evaluation and Automatic Program Generation*, Prentice Hall International, UK, June, 1993.

[103] G. Kahn, "The semantics of a simple language for parallel programming," *Information Processing 74: Proceedings of the Sixth IFIP World Computer Congress*, ed. J.L. Rosenfeld, Elsevier Science Publishers, 1974, pp 471–475.

[104] G. Karsai, "Design Tool Integration: An Exercise in Semantic Interoperability," *Proceedings of the IEEE Engineering of Computer Based Systems*, Edinburgh, UK, March, 2000.

[105] G. Karsai, G. Nordstrom, A. Ledeczi, and J. Sztipanovits. "Specifying Graphical Modeling Systems Using Constraint-based Metamodels," *IEEE Symposium on Computer Aided Control System Design*, Conference CD-Rom, Anchorage, Alaska, September 25, 2000.

[106] S. Kirkpatrick, and B. Selman, (1994), "Critical behavior in the satisfiability of random Boolean expressions," *Science* 264, 1297–1301.

[107] T. J. Koo, "Hierarchical system architecture for multi-agent multi-modal systems," In *Proc. of 40th IEEE Conference on Decision and Control*, 2001.

[108] A. Kurzhanksi and P. Varaiya, "Dynamic optimization for reachability problems," *J. Optimization Theory and Applications*, 108(2): 227–251.

[109] G. Lafferiere, G. Pappas and S. Sastry, "O-Minimal Hybrid Systems," *Mathematics of Control, Signals and Systems*, Vol. 13, pp. 1-21, 2000.

[110] T. Larrabee, and Y. Tsuji, "Evidence for satisfiability threshold for random 3CNF formulas," *Proceedings of the AAAI Symposium on Artificial Intelligence and NP-Hard Problems*, 112, 1993.

[111] J. R. Larus, S. K. Rajamani and J. Rehof. *Behavioral Types for Structured Asynchronous Programming*, Microsoft Research Technical Report, November 2001.

[112] A. Ledeczi, A. Bakay, and M. Maroti. "Model-Integrated Embedded Systems," in Robertson, Shrobe, Laddaga (eds), *Self Adaptive Software*, Springer-Verlag Lecture Notes in CS, #1936, February, 2001.

[113] A. Ledeczi, A. Bakay, M. Maroti., P. Volgyesi, G. Nordstrom, J. Sprinkle, and G. Karsai, "Composing Domain-Specific Design Environments," *Computer*, pp. 44-51, November, 2001.

[114] E. A. Lee, *Overview of the Ptolemy Project*, Technical Memorandum UCB/ERL M01/11, University of California, Berkeley, March 6, 2001.

[115] Edward A. Lee, "Embedded Software," to appear in *Advances in Computers* (M. Zelkowitz, editor), Vol. 56, Academic Press, London, 2002.

[116] E. A. Lee and D. G. Messerschmitt, "Synchronous Data Flow," *Proc. of the IEEE*, September, 1987.

[117] E. A. Lee and D. G. Messerschmitt, "Engineering an Education for the Future," *IEEE Computer Magazine*, Vol. 31, No. 1, January, 1998.

[118] E. A. Lee and T. M. Parks, "Dataflow Process Networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773-801, May, 1995.

[119] E. A. Lee and Y. Xiong, "System-Level Types for Component-Based Design," *Proc. of EMSOFT 2001*, Tahoe City, CA, *LNCS 2211,* Springer-Verlag, *October*, 2001.

[120] E. A. Lee and P. Varaiya, "Introducing Signals and Systems—The Berkeley Approach," *Proc. of the First Signal Processing Education Workshop*, Hunt, Texas, October 15 - 18, 2000.

[121] T. Levendovszky, G. Karsai, M. Maroti, A. Ledeczi, and H. Charaf. "Model Reuse with Metamodel-based Transformations," accepted for publication at *ICSR-7*.

[122] J. Liu, J. Eker, J. W. Janneck and E. A. Lee, "Realistic Simulations of Embedded Control Systems," to be presented at the International Federation of Automatic Control, 15th IFAC World Congress, Barcelona, Spain, July 21-26, 2002.

[123] J. Ludvig, J. McCarthy, S. Neuendorffer, and S. R. Sachs , "Reprogrammable Platforms for High-speed Data Acquisition," Submitted to *Design Automation for Embedded Systems*.

[124] J. Lygeros, D. N. Godbole, S. Sastry, "Verified Hybrid Controllers for Automated Vehicles," *IEEE Transactions on Automatic Control*, Vol. 43, No. 4, pp. 522-539.

[125] J. Lygeros, C. Tomlin and S. Sastry, *An Invitation to Hybrid Systems*, *monograph in preparation*, to be published by Springer Verlag in 2004.

[126] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1996.

[127] N.A. Lynch and B. Krogh (eds), *Proceedings of the Third International Workshop on Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1790, Springer-Verlag, 2000.

[128] K.L. McMillan, "A Compositional Rule for Hardware Design Refinement," *Computer-aided Verification*, 1997, Springer-Verlag, pp 24–35, Lecture Notes in Computer Science 1254.

[129] D. Messerschmitt, and C. Szyperski, *Industrial and Economic Properties of Software: Technology, Processes, and Value*, University of California at Berkeley Computer Science Division Technical Report UCB//CSD-01-1130, Jan. 18, 2001, and Microsoft Corporation Technical Report MSR-TR-2001-11, Jan. 18. 2001.

[130] Microsoft Corporation, *Universal Plug and Play Device Architecture*, June 2000. Available at http://www.upnp.org/download/UPnPDA10_20000613.htm, Jan. 4, 2002.

[131] R. Milner, *Communication and Concurrency*, Prentice-Hall, 1999.

[132] J.C. Mitchell, Foundations for Programming Languages, MIT Press, 1996.

[133] D. Mitchell, B. Selman, and H. Levesque, (1992), "Hard and easy distributions of SAT problems," Proc. 10th Natl. Conf. on Artificial Intelligence, AAAI Press, Menlo Park, CA, 459–465.

[134] *Model-based Synthesis of Generators for Embedded Systems*, online at http://www.isis.vanderbilt.edu/Projects/MoBIES/.

[135] *Modelica and the Modelica Association*, online at http://www.modelica.org/.

[136] P. J. Mosterman, G. Biswas and J. Sztipanovits, "Hybrid Modeling and Verification for Embedded Control Systems", *Control Engineering Practice*, vol. 6, pp. 511-521, 1998.

[137] P. J. Mosterman and G. Biswas, "Diagnosis of Continuous Valued Systems in Transient Operating Regions," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 29, pp. 554-55, 1999.

[138] P. J. Mosterman and G. Biswas, "Physical Semantics for Formal Verifications of Hybrid Systems Models", *Artificial Intelligence Journal*, Vol. 121, pp. 171-209, 2000.

[139] P. J. Mosterman and G. Biswas, "Towards Procedures for Deriving Hybrid Models of Complex Systems," *Hybrid Systems: Computation and Control*, LNCS Vol. 1790, pp. 324-337, 2000.

[140] E. Najm and A. Nimour, "Explicit Behavioral Typing for Object Interface," *Semantics of Objects as Processes*, ECOOP'99 Workshop, Lisboa, Portugal, June, 1999.

[141] G. Pappas, G. Lafferiere and S. Sastry, "Hierarchically Consistent Control Systems", *IEEE Transactions on Automatic Control*, Vol. 45, pp. 1145-1160, 2000.

[142] J. Preu, S. Kowalewski, H. Wong-Toi, and T.A. Henzinger. "An algorithm for the approximative analysis of rectangular automata." In A.P. Ravn and H. Rischel, editors, *FTRTFT 98: Formal Techniques in Real-time and Fault-tolerant Systems*, Lecture Notes in Computer Science 1486, pages 228–240. Springer-Verlag, 1998.

[143] F. Puntigam, "Types for Active Objects Based on Trace Semantics," *Proc. of the Workshop on Formal Methods for Open Object-Oriented Distributed Systems (FMOODS'96)*, Paris, France, March, 1996.

[144] H. J. Reekie, *Toward Effective Programming for Parallel Digital Signal Processing*, Research Report 92.1, University of Technology, Sydney, PO Box 123, Broadway NSW 2007, May 1992.

[145] M. D. Rice and S. B. Seidman, "A formal model for module interconnection languages," *IEEE Transactions on Software Engineering*, Volume: 20 Issue: 1 , Jan. 1994 pp: 88–101.

[146] *Rosetta System-Level Design Language*, online at http://www.ittc.ku.edu/Projects/rosetta/index.html.

[147] G. Rozenberg (ed.): Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations, World Scientific, 1997.

[148] J. Rumbaugh and G. Booch and I. Jacobson, *The UML Reference Guide*, Addison Wesley Longman, 1999.

[149] A. Sangiovanni-Vincentelli, "Defining Platform-Based Design," *EEDesign Exclusive Feature,* March 16, 2002. Available at http://www.eedesign.com/features/exclusive/OEG20020204S0062.

[150] S. S. Sastry and M. Cohn and F. Tendick, "Millirobotics for Remote, Minimally-Invasive Surgery," *Journal of Robotics Systems*, 1997.

[151] B. Selic, G. Gullekson, and P. Ward, *Real-Time Object-Oriented Modeling*, John Wiley & Sons, New York, NY 1994.

[152] *Semantic Designs Inc*., online at http://www.semdesigns.com/.

[153] O. Shakernia, G. Pappas and S. Sastry, "Semi-decidable synthesis for triangular hybrid systems," *Procedings of the 4th International Workshop on Hybrid Systems; Computation and Control, Rome,* 2001.

[154] *Sharable Content Object Reference Model*, at http://www.adlnet.org/.

[155] D. R. Smith: "Mechanizing the Development of Software," *Calculational System Design, Proceedings of the International Summer School Marktoberdorf*, M. Broy (Ed.), NATO ASI Series, IOS Press, Amsterdam, 1999.

[156] G. C. Smith, R. L. Clark and K. D. Frampton, "Optimal Transducer Placement for Active Control of Sound Transmission through Aeroelastic Plates," *Journal of Intelligent Material Systems and Structures*, Vol. 9, No. 12, December, 1999.

[157] *The Stanford SUIF Compiler Group* , online at http://suif.stanford.edu/.

[158] J. Stoy, Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory, MIT Press, Cambridge, MA, 1977.

[159] Sun Microsystems, *Jini™ Technology Architectural Overview*, Jan. 1999. Available at http://www.sun.com/jini/whitepapers/architecture.html.

[160] S. Swan, *An Introduction to System Level Modeling in SystemC 2.0*, Cadence Design Systems, Inc., draft report, May 2001.

[161] C. Szyperski, Component Software—Beyond Object-Oriented Programming. Addison-Wesley, 1998.

[162] C. Szyperski, "Rethinking our trade and science: from developing components to component-based development". *Modular Programming Languages. Joint Modular Languages Conference, JMLC 2000*. Zurich, Switzerland, 6-8 Sept. 2000.

[163] F. Tendick and S. Sastry and R. Fearing and M. Cohn, "Applications of Micro-Mechatronics in Minimally Invasive Surgery," *IEEE/ASME Transactions on Mechatronics*, February 1997.

[164] F. Tendick, M. C. Çavuşoğlu, "Human Machine Interfaces for Minimally Invasive Surgery," pp 2771–2776, Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS '97), October 1997.

[165] F. Tendick and M. S. Downes and M. C. Çavuşoğlu and L. W. Way, "Development of Virtual Environments for Training Skills and Reducing Errors in Laparoscopic Surgery," pp 36–44, *Proceedings of the SPIE International Symposium on Biological Optics (BIOS '98)*, January 1998.

[166] C. Tomlin, Hybrid Control of Air Traffic Management Systems, PhD thesis, UC Berkeley, 1998.

[167] C.J. Tomlin, J.L. Lygeros and S.S. Sastry, `"A game theoretic approach to controller design for hybrid systems," *Proceedings of the IEEE*, Vol. 88, pp. 949–970, 2000.

[168] C. J. Tomlin, J. L. Lygeros and S. S. Sastry, "Synthesizing Controllers for Nonlinear Hybrid Systems," *Hybrid Systems, Computation and Control*, Springer Verlag Lecture Notes in Computer Science, eds T. Henzinger and S. Sastry, pp. 360-373, 2000.

[169] J. Tsay, C. Hylands and E. A. Lee, "A Code Generation Framework for Java Component-Based Designs," *CASES '00*, November 17-19, 2000.

[170] University of California at Berkeley, *EECS20 – Structure and Interpretation of Signals and Systems*, on-line course notes at http://ptolemy.eecs.berkeley.edu/eecs20/.

[171] F.W. Vaandrager and J.H. van Schuppen (eds), *Proceedings of the Second International Workshop on Hybrid Systems—Computation and Control*, Lecture Notes in Computer Science 1569, Springer-Verlag, 1999.

[172] *VaNTH ERC in Bioengineering Educational Technologies*, online at http://www.vanth.org/.

[173] R. Vidal, S. Shaffert, J. Lygeros, and S. Sastry, "Controller Invariance of Discrete Time Systems," *Hybrid Systems: Computation and Control: 3rd Intl. Workshop (HSCC 2000)*, Pittsburgh, PA, edited by B. Krogh and N. Lynch, pp. 437-450, 2000.

[174] G. Winskel, The Formal Semantics of Programming Languages, MIT Press, 1993.

[175] W. Wolf, "What is embedded computing?", *Computer* , Volume: 35 Issue: 1 , Jan 2002, pg 136–137.