

Physical/Information Co-Design

Bill Mark

SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
(650) 859-4530
bill.mark@sri.com

ABSTRACT

Pervasive computing requires a new kind of software design, in fact a new kind of co-design. Pervasive system design must incorporate the characteristics and constraints of a dynamic physical environment. The critical constraint of the physical world are defined in terms of space and time; and the complexity of the real world often requires uncertainty modeling. The pervasive computing systems situated in this world are highly dynamic in terms of both composition and tasking. Pervasive computing software design requires the specification and implementation of a dynamic architecture in terms of physical, electronic, and software constraints – a co-design task. The co-design environment we are building combines our leading-edge symbolic system design/analysis technology with links to implementation in an open agent architecture. The emphasis is on the specification and implementation of architectures in terms of dynamically changing physical constraints.

INTRODUCTION

Almost all current software design is highly abstracted from the computational environment that will execute it. Indeed, that is one of the great accomplishments of computer science. Drivers, operating systems, and compilers provide layers of abstraction that free the designers from the need to understand the underlying hardware architecture (see Figure 1). The hardware architecture itself is fixed; in fact, a side-effect of this paradigm is that the daunting cost of creating and maintaining the abstraction layers makes the introduction of new hardware architectures very rare.

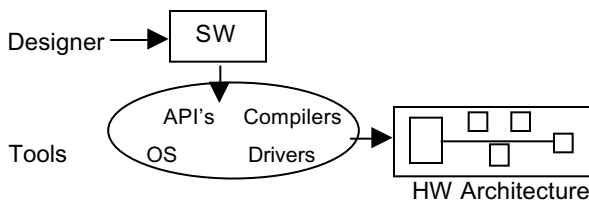


Figure 1: Conventional software design

HARDWARE/SOFTWARE CO-DESIGN

However, for certain problems, hardware architecture must be seen as a variable of the design task. Hard real-time constraints and the need for power conservation require a

completely different design paradigm: hardware/software co-design (see Figure 2).

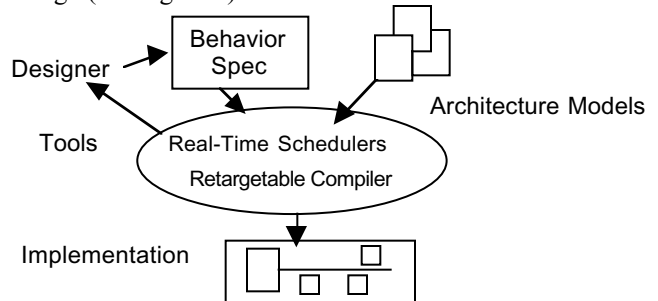


Figure 2: Hardware/software co-design

To achieve required performance with minimal hardware resources, designers must create an architecture that takes into account – ideally is optimized for – the computational task. The co-design environment provides tools (schedulers, retargetable compilers, etc.) for linking the specification of system behavior to constraints on the hardware architecture. Different choices of algorithms, communication among functional blocks (software or hardware), and allocation of resources to general versus special purpose hardware can be used to create designs that achieve the desired balance of execution speed, power consumption, and cost.

Most current co-design work is directed toward cost/performance-optimal design of narrow classes of real time systems. To make implementation of the design environment manageable, the actual variability of the architecture is fairly restricted, usually to parameterizations of a single underlying platform. For example, even the more advanced commercial co-design environments (e.g., Cadence VCC [1], Coware N2C [2]) allow variability only within a small class of processors and bus architectures.

Recent research work (e.g., [3, 4]) is looking more generally at the problem of designing systems that incorporate multiple computational models. The approach described here is a further step in that direction, based on the specific computational characteristics of pervasive computing.

COMPUTATIONAL CHARACTERISTICS OF PERVASIVE COMPUTING

Conventional computer platforms obviously occupy physical space, but this is usually irrelevant. Few software

designers need to care about the distances and spatial layouts of their computing platforms. For multiple platform computation, software designers are encouraged to disassociate computation from location: computation is available from any tap, “the network is the computer”. While this is a valuable viewpoint that will certainly continue in the pervasive computing world, it is based on the separation between computers and real things. In the pervasive computing world, the network will still be the computer, but the computer will be a set of communicating situated computational nodes. The set may be very large and dynamically changing, the communication may be low bandwidth and unreliable, and the nodes may be relatively simple.

Situated Nodes

A computer is an artificial entity; it does not matter very much where it is, especially in a networked world. This is very different from a computational element that is embedded in a physical environment. Many embedded computational elements will be inseparable from specific physical objects, not only in the sense of physical identity, but also in the sense of task identity (e.g., a computational pencil). While the computational pencil may have other functionality, it must at least have pencil functionality at all times [5,6].

Dynamic, Uncertain Networks

Most pervasive computation will require the interaction of more than one – perhaps very many – nodes. The network that results from inter-node communication is thus a key computational characteristic of pervasive computing.

Pervasive computing networks will almost always be dynamic because people and devices move around in space, changing position and focus, coming and going. In addition, many pervasive computing environments will rely on wireless communication, making the network both dynamic and uncertain (nodes will drop out and reappear; bandwidth limitations may require dynamic reconfiguration).

PHYSICAL/INFORMATION CO-DESIGN

Pervasive computing thus shares some of the computational characteristics that drive hardware/software co-design, including the need for real-time and power-constrained design. But the greatest resonance with the hardware/software co-design paradigm is that in pervasive computing system design, the system behavior and the architecture that will implement that behavior must be co-designed: one cannot be determined without the other.

In the pervasive computing world, “the system” is defined dynamically in terms of:

- available resources (the set of computational elements that are within a space, within a power/bandwidth budget, etc.);
- task (the computation required at a given moment to meet the users’ needs); and
- environment (situation-dependent communication and sensor characteristics).

The collection of nodes that comprise the system thus changes frequently. The architecture that *turns those nodes into a system* must be a dependent variable of the design, since it is a function of other factors that are chosen at design time (node interaction protocols, algorithms chosen to perform tasks, etc.). Furthermore, this dynamic architecture must be actively managed at runtime – the architecture manager is a major output of the design process (see Figure 3).

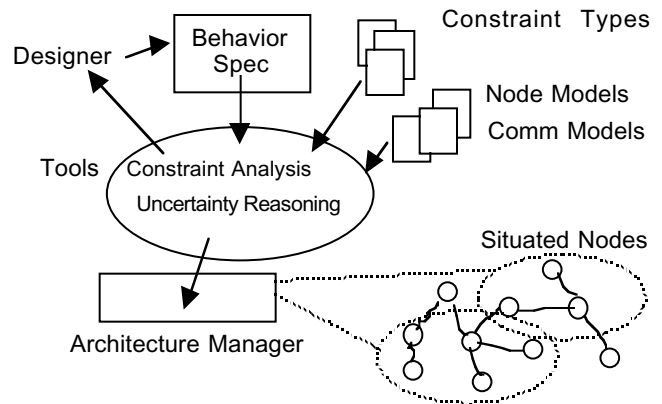


Figure 3: Physical/information co-design for pervasive computing

At SRI, we are working on an environment to co-design pervasive computing systems in terms of the physical and information constraints that define system behavior.

Designing in Terms of Invariants

The key to creating a reliable system in a changing world is to state the design in terms of things that do not change: the invariant constraints of the system.

Some of the key invariants for designing pervasive systems are:

temporal: system behavior is invariant with respect to the specific time of component execution within a given interval;

spatial: system behavior is invariant with respect to the specific location of component execution within a given volume;

information flow: system behavior is invariant with respect to the order in which components receive information;

type: system behavior is invariant with respect to the choice of specific components within a given type.

These constraints (and others – this list is just suggestive) clearly interact, possibly in complex ways. The co-design environment must help the designers by computing the interaction of these constraints and giving the designer feedback about changes. The PVS system developed at SRI [7,8] has leading-edge capability for specifying symbolic constraints and computing their ramifications (in fact, proving properties about systems of constraints).

System behavior will also depend on node and communication behavior that may need to be modeled probabilistically. The co-design environment must help

the designer to integrate these models into the symbolic constraints that specify higher-level system behavior.

Links to Implementation

In any co-design environment, the design must be linked to implementation, hopefully automatically. In hardware/software co-design, this link is created by compilers and hardware description language generators. For pervasive computing, the co-design environment must generate a runtime component that manages the dynamic architecture.

Open Agent Architecture (OAA) [9] provides an appropriate vehicle for dynamic run-time architecture management. Its flexible agent communication language and agent facilitation structure allow agents (representing pervasive computing nodes and services) to be gathered and managed dynamically to form a system and achieve specified higher-level behavior.

Once the designers arrive at a set of constraints and models that specify the desired system behavior, the co-design environment generates the required OAA structure. This consists of wrappers to allow nodes to dynamically register their capabilities and a mediator (a specialized agent) to break higher-level tasks into sub-tasks that can be executed by registered agents.

STATUS

The concept for this new co-design environment is based on highly developed SRI code-and-idea platforms (PVS and OAA). Creating the new environment requires further development and integration of these platforms:

- an interface to PVS that enables designers to manipulate specifications and receive interactive feedback on the results of constraint analysis;
- better techniques for expressing uncertainty; and
- generators that produce OAA code from PVS expressions.

In addition, the environment must be populated with template building blocks for pervasive computing system specifications:

- a library of symbolic expressions for the invariants described above; and
- node models.

The initial phase of these developments is now underway at SRI.

ACKNOWLEDGEMENTS

These ideas build on the work of many researchers at SRI, in particular Patrick Lincoln, Victoria Stavridou, and Charles Ortiz.

REFERENCES

1. <http://www.cadence.com/features/vol15No1/VCC.html>
2. <http://www.coware.com/proinfo.html>
3. Lee, E., "Embedded Software – A Research Agenda", UCB ERL Memorandum M99/63, December, 1999.
4. Chou, P. and G. Borriello, "An Analysis-Based Approach to Composition of Distributed Embedded Systems," in the *Proceedings of CODES'98 (International Workshop on Hardware/Software Codesign)*, March 1998, Seattle, WA. pp.3-7.
5. Mark, W., "Turning Pervasive Computing into Mediated Spaces", *IBM Systems Journal*, Vol. 38, No. 4, 1999, pp. 677 – 692.
6. Norman, D., *Things That Make Us Smart*, Perseus Books, Reading, MA (1993).
7. <http://pvs.csl.sri.com/>
8. Saidi, H. and N. Shankar, "Abstract and Model Check While You Prove", in *Computer-Aided Verification (CAV '99)*, Trento, Italy, July 1999. *Springer-Verlag Lecture Notes in Computer Science*, Vol. 1633.
9. Martin, D., *et al.*, "The Open Agent Architecture: A Framework for Building Distributed Software Systems", *Applied Artificial Intelligence: An International Journal*, Volume 13, Number 1-2. January-March 1999.