

Embedded Reasoning

Model-based design for highly
reconfigurable systems

Contact:

Jennifer Ernst
Director, Business Development
Palo Alto Research Center Inc.

+1 (650) 812-4916
jennifer.ernst@parc.com



Copyright © 2006 Palo Alto Research Center Incorporated.
All Rights Reserved.

PARC and the PARC Logo are trademarks of Palo Alto
Research Center Incorporated.

Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304 USA

www.parc.com
info@parc.com
+1 (650) 812-4000

Motivation

Hardware and software systems are becoming increasingly complex. Whether considering the software-enabled car, the smart home, the reconfigurable factory – the number of embedded sensors, actuators, and independent processors is growing rapidly.

A central challenge in designing such systems is the complexity of components and large number of possible system configurations. It is difficult and, in many cases, virtually impossible to anticipate every configuration and contingency when developing the control software. Control software development is increasingly the leading cause for project time and cost overruns, and thus can be a major barrier to deploying advanced systems capabilities.

Model-based systems avoid hard-coding pre-defined behaviors. Instead, the system is designed to reconfigure itself dynamically for the current goal. The control system uses a model of the capabilities and constraints of individual components to reason about overall system behavior.

With this approach, the system autonomously:

- infers its overall capabilities (from constraint-based models of individual components);
- computes the optimal behavior for the specific task;
- plans what to do and schedules when to do it;
- configures components just in time and controls their actions;
- continually monitors and diagnoses itself;
- requests upgrades when necessary;
- dynamically reacts to changes (e.g., re-routing a process within milliseconds if something suddenly breaks).

PARC Research Directions

Model-based design begins with developing declarative models that provide a generic representation of a system and its components. Explicit objectives can then be identified. Intelligent systems compute optimal behaviors and coordinate components to accomplish the desired outcomes.

PARC's research encompasses three major trajectories necessary for robust intelligent system behavior—reflection, adaptation, and embedding.

Reflection, the explicit representation of and reasoning about the system and its constraints, can dramatically simplify development and increase the capabilities of the control software and thus the overall system. In a reflection-based approach to control, models describe the structure, behavior, goals, and constraints of a system's

components. From these models, the software can determine how to achieve the system's goals given its capabilities, it can robustly coordinate component interactions, and it can determine the state and health of the components. The power and flexibility of model-based approaches will lead to robust control systems with higher functionality developed in a shorter time frame.

Adaptation is a system's reaction to dynamic goals, changing components, and even changing configurations. PARC's approach goes further than many model-based approaches by ensuring the system is not only "intelligent", but also self-aware. Mechanisms used for adaptation include planning, scheduling, control, performance optimization, action selection, environment understanding, fault tolerance, and user interaction. This involves online self-monitoring, automated diagnosis, and real-time planning and scheduling. The resulting system can configure components just in time, request upgrades, and dynamically respond to changes. Reflection-based software using explicit on-line models is particularly suitable for adaptive systems, as models can be changed easily and even be learned and modified automatically.

Embedding software in distributed systems often implies real-time algorithms with timeliness or quality guarantees, integrated architectures, and distributed autonomous or semi-autonomous implementations. Many advanced techniques have been developed in the control, artificial intelligence, and robotics communities that enable reflection and adaptation, including model-based reasoning, model-predictive control, constraint-based planning and scheduling, offline and online learning, intelligent task and resource allocation, and multi-agent negotiation. Few techniques, however, are ready to be integrated with each other and to be embedded in real-time, resource-constrained, and distributed systems. Embedding will require both novel approaches and careful engineering to deliver the promise of software-enabled control.

Benefits

The adaptive, model-based approach provides a number of benefits.

- less manual programming
- provable properties (correctness, optimality)
- new control capabilities (adaptive software, use of redundancy, generic fault recovery)
- increased reuse of code
- improved software engineering (modular architecture, standard interfaces, re-use)
- significantly reduced time to market
- simpler user interfaces for the system operator
- increased reliability and performance by adapting to internal and external changes.

Case: Intelligent Control for Xerox Print Systems

A printer is a complex electro-mechanical multi-function device that processes images, moves sheets of paper at 1-3 meters/second, and typically has a variety of feeding, marking, and finisher options comprised of hundreds of components. The interactions between these operations can be quite complex.

It's the job of the control software to make sure that documents are printed as fast as possible without jams and other errors. Hard-coding all possible system behaviors would be tremendously difficult and impractical.

Xerox Corporation wanted to offer customers a way to customize their printing systems and flexibly add capabilities as their business needs change. This implied that, beyond their inherent complexity, systems would also be changing over time.

PARC researchers designed control software that uses a generic reasoning engine and modular architecture to let the printing system:

- coordinate complex, distributed, and often networked components;
- adapt to new components or customer plug-ins without having to manually retool the software;
- ensure maximum productivity, reliability, and performance with minimal disruption.

This approach uses explicit system knowledge, in the form of constraint-based models of the components. The "generic" control software provides techniques for representing the capabilities of system components, and for inferring from those the capabilities of the overall system. Descriptions of the system configuration – for example, a multi-function device with multiple finishing options – are provided, along with a series of goals (i.e., print jobs) to be achieved. Given the configuration descriptions, embedded real-time planning and scheduling algorithms automatically select the capabilities that will correctly and optimally achieve the goals. The component models are reusable for different machine configurations and for different tasks such as fault recovery.

PARC's "universal" approach met the requirement for Xerox customers' to be able to grow their systems over time. Equally as important, however, it enabled Xerox to unify its software base. The resulting software now drives multiple Xerox product families including the flagship iGen3™ digital production printing system.

PARC researchers continue to enhance this technology for Xerox printing systems. Enhancements include control software for recovering from faults, dynamically rerouting jobs, and clearing the system if a jam occurs. The software also enables the system controller to adapt dynamically to equipment being taken off-line in the field, maximizing throughput with the current configuration.