Cost-Performance Tradeoff for Embedded Systems

Julie S. Fant and Robert G. Pettit

The Aerospace Corporation
Chantilly, Virginia USA
{julie.s.fant,robert.g.pettit}@aero.org

Abstract. Software engineering requires creativity, thorough design and analysis, and sound design decisions. Design decisions often have tradeoffs and implications associated with them. Therefore, it is important that design decisions are based on sound analysis. With respect to embedded systems, key drivers are often performance and cost. Thus the purpose of this paper is to describe an approach to aid in the design decision process on cost and performance tradeoffs for embedded systems. Specifically, it presents a model-driven approach to understand and communicate the performance-cost tradeoff.

Keywords: software performance, cost, UML, embedded systems, model-driven design, tradeoff.

1 Introduction

Software engineering, like other engineering disciplines, requires creativity, thorough design and analysis, and sound design decisions. Design decisions often have tradeoffs and implications associated with them. Thus these decisions should be examined with proper analysis to ensure the best overall decision is made.

Embedded systems are a special type of system where the computers and associated software are components embedded within a larger system such as mobile phones, household appliances, automotive controls, etc. In these types of systems, performance and cost are often key drivers. Many times the solution to achieving better performance is simply to purchase more expensive hardware. This, however, is not always a good solution since the additional cost of the high performance hardware may not result in an equivalent performance gain. For example, one may spend a large sum of money for the fastest central processor available but find that input/output (I/O) constraints limit the benefits from the high-performance CPU. Therefore it is critical to spend the time analyzing the different options to ensure the best decision is made between cost and performance.

The purpose of this paper is to describe an approach to aid in the design decision process by helping to understand and communicate the performance-cost tradeoff for embedded systems. Specifically, it presents a model-driven approach that combines software performance analysis techniques with techniques to analyze and compare the cost-performance aspects of potential hardware implementations.

This paper is structured as follows: Section 2 describes the related works. Section 3 presents the approach to cost-performance analysis and its benefits. Section 4 describes a case study using the proposed approach. Finally, Section 5 contains the major conclusions and future work.

U. Brinkschulte, T. Givargis, and S. Russo (Eds.): SEUS 2008, LNCS 5287, pp. 198–208, 2008.
© IFIP International Federation for Information Processing 2008

2 Related Work

Many approaches to analyze the performance of embedded and real-time systems have been developed. For our purposes, these approaches can be broadly categorized as those exploring performance through analytical techniques [1-6] or through simulation [7-10]. The cost-performance analysis approach presented in this paper does not prescribe the use of a particular performance analysis method. Rather, it attempts to illustrate how cost-performance tradeoff decisions can be compartmentalized and input to analytical or simulation techniques that will assist the decision making process.

3 Analysis Approach and Benefits

The paper presents an approach to performing cost-performance tradeoff analysis for embedded systems. The purpose of this approach is to help communicate and understand the cost-performance tradeoffs associated with different hardware implementation options. It has five major steps, which are as follows: 1) Develop a platform independent model; 2) Select the hardware configurations to analyze; 3) Conduct performance analysis on each of the hardware configurations; 4) Perform cost-performance tradeoff analysis; and 5) Make and document the design decision. Each step is described below in more detail.

The first step in the proposed tradeoff approach is to build a platform independent model of the software system. The purpose of this step is to show how the software is meeting the functional requirements. Additionally the platform independent model will serve as the foundation for predicting software performance. It is recommended that the models be captured using the Unified Modeling Language (UML) since it is the de facto object oriented modeling language in industry.

The next step is to select the hardware implementation options for the software system. A good way to promote creativity and to enumerate the different potential options is to develop morphological box. A morphological box is an existing systems engineering technique that uses a two-dimensional table of components and physical architecture options, as depicted in Table 1.

Table 1. Morphological Box Generic Example

Component A	Component B	Component C
Physical Option A1	Physical Option B1	Physical Option C1
Physical Option A2	Physical Option B2	Physical Option C2
Physical Option A3	Physical Option B3	

Each column represents a component and each row in the column represents a physical instantiation option. Different system physical architectures can be analyzed by selecting one box from each column [11]. This same technique can be applied to software for determining and selecting hardware implementation options. The columns will represent hardware elements and the rows will represent different physical hardware options for the elements. For example, one column may be the microcontroller with possible options being an H8 or an ARM 7 microcontroller. Once the

morphological box is created, the engineers can select the hardware implementations to analyze by selecting one row from each column. The morphological box will likely produce a large number of potential options. However, not all of these options need to be analyzed. Engineers should only select a subset that they are considering for the end system. Selections can be made with certain characteristics in mind such as lowest cost hardware or highest performance hardware.

Once the potential hardware implementations have been identified, the third step is to perform software performance analysis for each implementation. Any software performance analysis technique can be used in the proposed approach. For example, the UML platform independent model can be annotated with platform specific information using a UML profile and then subsequently analyzed. Alternatively, the UML platform independent model can be converted into a Petri-net model and subsequently analyzed for performance. The performance metrics produced in the software performance analysis should coincide with the software performance requirements. For example, if the system has a requirement for a maximum latency, then latency should be calculated in the performance analysis.

The fourth step in the proposed tradeoff approach is to compare the different hardware implementations against cost and performance. This should be done by developing tradeoff x-y scatter plots of performance and cost. The plots should again be based on the performance requirements. This will clearly show the tradeoff of different hardware configuration options on one graph. For example, if there is a performance requirement on the maximum latency, then the tradeoff x-y scatter plot should plot latency versus cost. Additionally, the performance requirements can also be added to the graph to show the system's threshold. To illustrate this point, consider Figure 1.

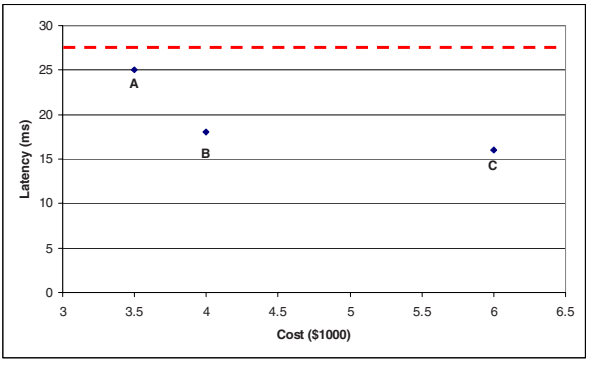

Fig. 1. Example Tradeoff Plot

This is an tradeoff x-y scatter plot of worst case latency versus cost for three hardware options and the performance requirement for maximum latency is denoted with a red-dotted line. In this example, all the options meet the performance requirement since they are below the maximum latency threshold. It can be clearly seen that there is a 28% increase in performance and a 14% cost increase between options A and B. Between options B and C there is an 11% performance increase, however the cost is

50% more. In this case, since option B provides the best balance between cost and performance, it is the best choice for the example system.

Finally, after the different options have been analyzed and a decision has been made, the design decision should be documented so that future maintainers of the system will understand why this decision was made.

The proposed tradeoff analysis approach has several benefits. First, the proposed tradeoff approach does not prescribe any particular performance analysis technique. This is good because it enables organizations to leverage their currently existing performance analysis techniques. Another benefit of the proposed tradeoff approach is that it provides an easy means to understand and communicate tradeoff decisions. The scatter plots present the data from all the potential hardware options on a single graph while illustrating the cost-performance impacts of each option. Finally, the proposed tradeoff approach helps directly link design decisions to performance requirements.

4 Case Study

In this section, we illustrate the cost-performance tradeoff approach using a robot controller case study. The robot controller is an autonomous robot with an infrared light sensor and two motors (actuators). The goal of the robot is to search an area for colored discs while staying within a course boundary and avoiding obstacles. In this case study, a light sensor is used as the sole input sensor, responsible for detecting boundaries, obstacles, and discs according to different color schemes. In order to avoid hitting obstacles and boundaries, the robot controller must process the light sensor inputs in a timely manner. For our purposes, the rover has a requirement to react to a light sensor event within a travel distance of 0.5 cm, which corresponds to 50ms in the configurations used for this study. The following subsections details each step in the proposed tradeoff approach.

4.1 Platform Independent Model

The first step in the tradeoff approach is to build a platform independent model of the robot controller to show how the system will meet its functional requirements. We designed the case study following the COMET method and stereotypes [5]. The system is divided into three active, concurrently executing objects (detect, rover, and nav), one passive object (map), and three external I/O objects for receiving light sensor input and for modeling output to the two motors. Figure 2 depicts a UML sequence diagram for how the different objects interact. The detect, rover, and nav objects all operate asynchronously and all messages between the active objects have synchronous, buffered communication.

4.2 Hardware Configuration Selection

The next step in the tradeoff approach is to develop the different hardware implementation options. In this example there are four hardware elements which are the two motors, the light sensor, and the microcontroller platform. In this configuration, the microcontroller platform performs all of the processing and the light sensor is the sole

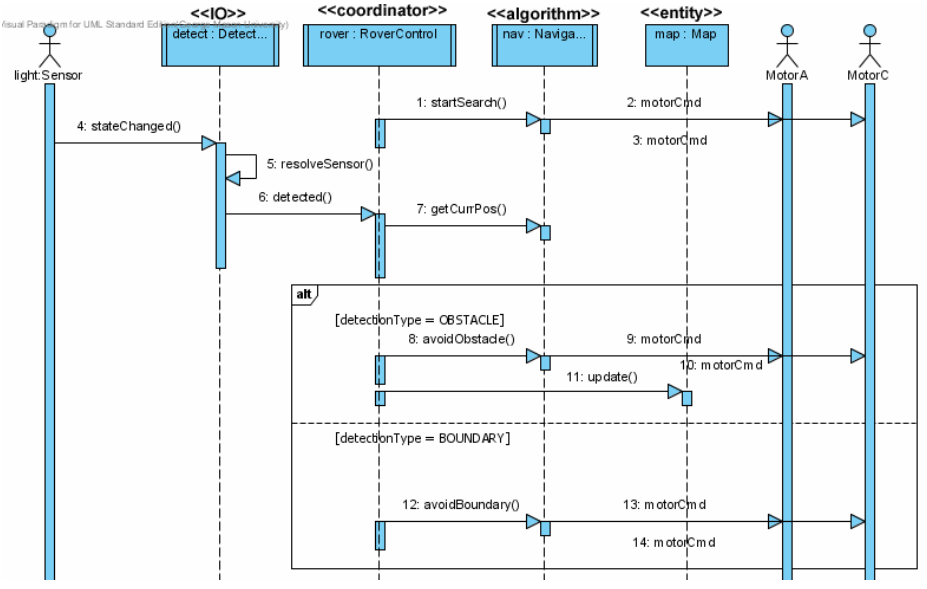

Fig. 2. Platform Independent Sequence Diagram

input for determining discs and obstacles. The microcontroller platform uses two motors which are used to maneuver the robot. Turning is achieved by rotating the left (Motor A) and right (Motor C) motors in opposite directions. These four elements become the columns in our morphological box and each element was given at least one hardware option.

The final morphological box for the robot controller is depicted in Table 2. In this example, we consider two types of motors, three different light detectors, and three different microcontrollers. These platform specific performance characteristics and costs were also listed in the morphological box. The performance characteristics were selected based on the notational embedded system framework described in [9]. This framework shows which platform characteristics need to be included in the design of concurrent software. We determined the platform specific characteristics and costs using online pricing, historical data, hardware specifications, and published benchmarks for the different systems [12-16].

After the morphological box is populated, it is time to select the hardware implementations that will be considered. In this example, we chose to analyze the cheapest option which is referred to as RP: two RCX interactive servo motors, the CDS photoresister, and the RCX Intelligent Brick. The second option we selected uses the highest performance hardware which is called JN: two NXT interactive servo motors, the NXT light sensor, and JOP. The third option we picked was the standard RCX configuration which is referred to as RR: two RCX interactive servo motors, the RCX light sensor, and the RCX Intelligent Brick. Finally, we also chose to analyze the new Mindstorms™ NXT system which is referred to as NN: two NXT servo motors, the NXT light sensor, and Mindstorms™ NXT processor.

Table 2. Morphological Box for the Robot Controller

Motor A	Motor C	Light Sensor	Platform
RCX Interactive Servo Motor Latency=1ms cost=$18	**RCX Interactive Servo Motor** Latency=1ms cost=$18	**Mindstorms™ Light Sensor** detectionLatency=10.3ms Cost=$17	**RCX Intelligent Brick - Hitachi H8 µ controller** IPS=18M clockspeed=16MHz csOverhead= < 1ms kbMemOverhead=17.5 RAM=28KB Cost=$45
NXT Servo Motor Latency=1ms cost=$18	**NXT Servo Motor** Latency=1ms cost=$18	**CDS Photoresister** detectionLatency=30ms Cost=$0.60	
		NXT Light Sensor detectionLatency=5ms Cost=$39	**JOP - Altera Cyclone EP1C6 FPGA Board** IPS=10406M clockspeed=20 MHz csOverhead= < 1ms kbMemOverhead=3KB RAM=92KBits Cost=$310.00
			Mindstorms™ NXT – ARM 7 µ controller IPS=80M clockspeed=40MHz csOverhead= <1 ms kbMemOverhead=20 RAM=64MB Cost=$135

4.3 Performance Analysis

The third step in the tradeoff analysis approach is to conduct the performance analysis. This is the step where the different hardware implementations are analyzed for performance. In this case study, to illustrate the flexibility of the cost-performance tradeoff approach, we will show the performance analysis using both an analytical and a simulation approach. The follow subsections show the details for each approach.

4.3.1 Analytical Technique

For an analytical technique, we start with a UML model augmented with platform specific characteristics and then apply event sequence analysis for certain performance scenarios. Here, platform specific UML models are annotated using the UML Profile for Schedulability, Performance and Time (SPT) [17]. The UML SPT profile is scheduled to be replaced by the UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [18], however the SPT profile is still adequate for the purposes of this paper.

Using this approach, we created a platform specific UML model for each of the hardware configurations being analyzed. At a minimum, the platform specific UML model must capture the hardware configuration in a deployment diagram and the processing steps in interaction diagrams such as a sequence diagram. Figure 3 shows the platform specific sequence diagram for the RCX Intelligent Brick with CDS photoresistor (RP) configuration. We estimated demand times for each step by dividing the number of estimated instructions per step by the microcontroller's IPS rate.

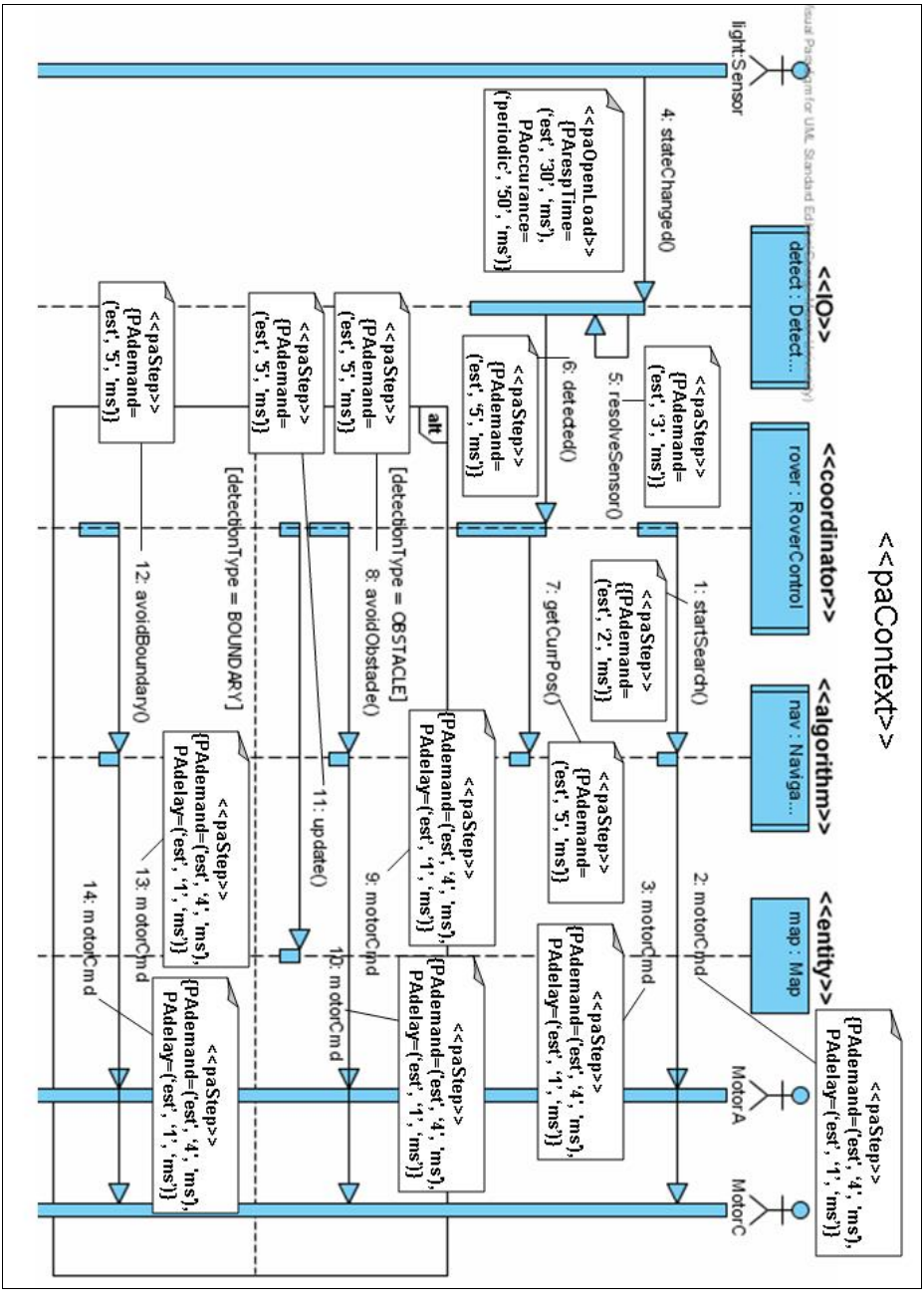

Fig. 3. Platform Specific UML Sequence Diagram for the RP configuration

After we created the platform specific UML models, we then performed event sequence analysis to determine the worst case latency through the system. Event sequence analysis is used to determine the tasks that need to be executed in order to service a given event. This is computed by calculating the time for the tasks in the event sequence plus any time used for context switching and message communication [5]. Table 3 provides a summary of the results for each of the configurations.

Table 3. Summary Performance Analysis Results

Short Name	Configuration	Worst Case Latency
JN	JOP w/NXT light sensor	6.1ms
RP	RCX Intelligent Brick w/photoresistor	50.7ms
RR	RCX Intelligent Brick w/RCX light sensor	31ms
NN	Mindstorms™ NXT w/NXT light sensor	10.5ms

4.3.2 Simulation

The simulation technique we used in this case study is simulation through coloured Petri nets(CPNs) by Pettit and Gomaa [7-9]. This method assigns behavioral patterns to the UML objects and constructs CPN templates for each behavioral pattern. Connecting the templates and populating with application and platform specific characteristics provides for an executable CPN model of the system that can be used to analyze such properties as throughput and concurrent behavior. Applying time-stamps to the tokens within the net also allows us to monitor the flow of events and messages over time and provides us with the capability to analyze response time (latency) from the receipt of an event to the output action associated with that event.

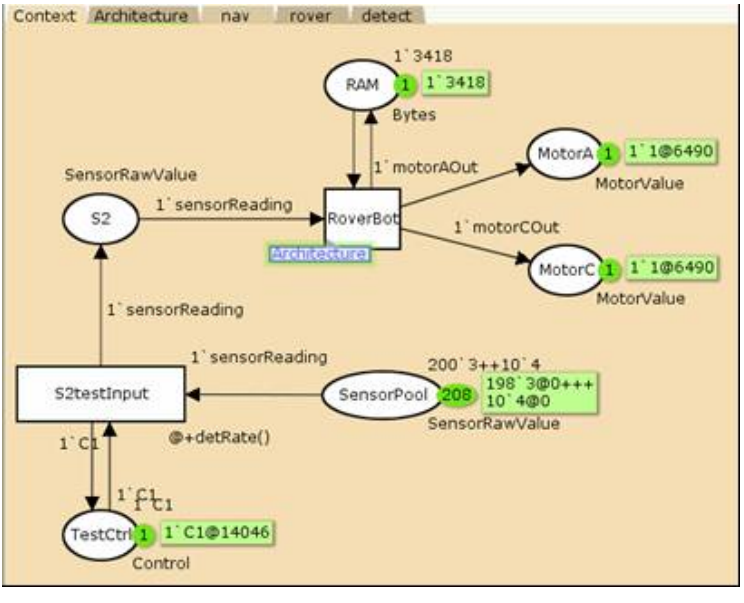

Fig. 4. CPN Simulated Response Time

Figure 4 shows the high-level results of simulating response times for the RR configuration. In this scenario, a light sensor event occurs at time 6459 (not shown on the figure) and a response to the motors is observed at time 6490 (simulation time in milliseconds). Thus, the reaction time for this case is 31ms. Further execution runs resulted in response times no greater than this value.

4.4 Cost-Performance Tradeoff Analysis

After we conducted performance analysis on the all the different hardware configurations, we created the cost-performance tradeoff plot. These plots can be derived from the analysis data, the simulation data, or both, depending on the availability of models and the desired confidence in the results. Figure 5 is the tradeoff plot of our performance analysis shows cost versus worst case latency. From this tradeoff plot we can see that the RP configuration does not meet the performance requirement; therefore it cannot be selected. We can also tell from the tradeoff plot that the lowest cost option that still meets the performance requirement is the RR configuration. The tradeoff plot also clearly shows that while the NN configuration does cost more (Δ$112), it does provide a significant performance increase (Δ20.5ms). We can also tell from this graph that the highest cost option, JN, does yield the fastest performance. However, this graph illustrates that the relative performance gain of Δ4.4ms between the NN and JN configuration probably does not outweigh the additional cost of Δ$175.

In summary, the tradeoff plot helps engineers in their design decision process. For this system, if the overall goal is to keep costs low, then the RR configuration is the best option since it is the lowest cost option that still meets the performance requirements. If the overall goal is to maximize performance while keeping costs down, then the NN configuration is the logical choice since it has a reasonable balance of cost and performance.

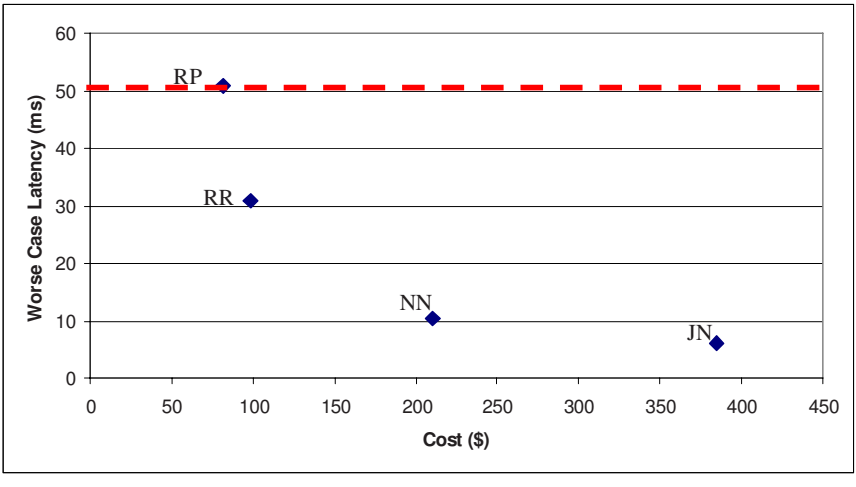

Fig. 5. Tradeoff plot for robot controller case study

5 Conclusions and Future Work

In conclusion, the proposed cost-performance tradeoff approach is intended to help in the design decision processes to ensure the best overall decision is made. Specifically, it helps to examine and illustrate the tradeoffs between cost and performance for embedded systems. This helps engineers ensure performance requirements are met and cost is considered in the processes. This helps avoid the unnecessary purchase of expensive hardware and helps keep the overall system cost low. The approach is also flexible enough to work with any software performance analysis technique which companies maybe using. This enables organizations to leverage the performance analysis technique already in existence. Finally, it provides an easy means to understand, communicate, and document tradeoff decisions. The tradeoff plots present the data from all the potential hardware options on a single graph which makes the data easy to communicate and understand.

A next logical extension of this approach would be to tradeoff decisions with other non-functional aspects of software such as security or reliability. For example, the approach can examine the performance impacts of including various security measures in a system. Tests should also be expanded to larger systems to prove scalability of the approach.

References

1. Wu, X., Woodside, M.: Performance modeling from software components. In: Proceedings of the 4th international workshop on Software and performance, Redwood Shores. ACM Press, California (2004)
2. Woodside, M., et al.: Performance by Unified Model Analysis (PUMA). In: Fifth International Workshop on Software and Performance (WOSP 2005), Palma, Illes Balears, Spain (2005)
3. Sabetta, A., et al.: Annotating UML Models with Non-Functional Properties for Quantitative Analysis. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 79–90. Springer, Heidelberg (2006)
4. Wu, X., McMullan, D., Woodside, M.: Component Based Performance Prediction. In: 6th ICSE Workshop on Component-Based Software Engineering: Automated Reasoning and Prediction, Portland, Oregon (2003)
5. Gomaa, H.: Designing Concurrent, Distributed, and Real-Time Applications with UML, 3rd edn. Addison-Wesley Object Technology Series, Boston (2000)
6. Street, J., Gomaa, H.: An Approach to Performance Modeling of Software Product Lines. In: 9th International Conference on Model Driven Engineering Languages and Systems Modeling and Analysis of Real-Time and Embedded Systems (MARTES) Workshop, Genova (2006)
7. Pettit, I.R.: Analyzing Dynamic Behavior of Concurrent Object-Oriented Software Design Ph.D Dissertation, in Department of Information and Software Engineering, George Mason University: Fairfax, VA (2003)
8. Pettit IV, R., Gomaa, H.: Modeling Behavioral Design Patterns of Concurrent Objects. In: Conference on Software Engineering (ICSE), Shanghai, China (2006)
9. Pettit IV, R., Gomaa, H.: Analyzing Behavior of Concurrent Software Designs for Embedded Systems. In: ISORC 2007. IEEE, Los Alamitos (2007)

10. Ober, I., Graf, S., Ober, I.: Validating timed UML models by simulation and verification. In: Workshop on SVERTS: Specification and Validation of UML models for Real Time and Embedded Systems, San Francisco, California, USA (2003)
11. Buede, D.: The Engineering Design of Systems Models and Methods. Wiley Series in Systems Engineering. Sage, Thousand Oaks (2000)
12. Performance of Various Java Processors (2006) [cited May 2008], http://www.jopdesign.com/perf.jsp
13. MINDSTORMS(R) - Legos Shop [cited May 2008], http://shop.lego.com/ByTheme/Leaf.aspx?cn=17&d=70
14. Macron Photoresistor Specification [cited May 2008], http://www.macron.com.hk/spec_photoersistor.htm
15. Radio Shack [cited May 2008], http://www.radioshack.com/sm-cds-photoresistorsassortment-of-5-pi-2062590_tb-techSpecs.html
16. Altera Online [cited May 2008], http://www.altera.com/
17. The UML Profile for Schedulability, Performance and Time (January 2005) cited, http://www.omg.org/technology/documents/formal/schedulability.htm
18. UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) (2007) cited, http://www.omg.org/cgi-bin/doc?ptc/2007-08-04