Proposal and Evaluation for Organising and Using Available Data for Software Performance Estimations in Embedded Platform Development

Enrico Johansson
Department of Communication Systems
Lund University
Box 118, SE-221 00 Lund, Sweden
enrico.johansson@telecom.lth.se

Fredrik Wartenberg
Ericsson Mobile Platforms AB
Nya Vattentorget, SE-221 83 Lund, Sweden
fredrik.wartenberg@ericsson.com

Abstract

Embedded platform development, where a platform is a basis for a family of products, is characterized by soft- and hardware co-design as well as very tight constraints on cost and available real-time system performance. The latter implies that early software performance estimation plays a crucial role in embedded platform development. Due to the different focus of software and hardware design, performance data and estimations techniques needed are by nature heterogeneous. In order to simplify access and availability of software performance data for an entire development organization, a database tool is proposed and implemented, which allows organizing, presenting, and to some extent evaluating the existing, inhomogeneous data for software performance estimations.

1 Introduction

Estimations of software performance [5, 6] refer to providing the response times and the workloads before the software is fully implemented and integrated in a product. Software performance is a critical design criterion, especially for embedded products with restricted resources and strong cost constraints [1, 8, 14]. In a large "embedded" company, like Ericsson Mobile Platforms AB (EMP) [4], work on software performance is carried out in many parts of the development projects. EMP is committed to supply phone manufacturers and original device manufacturers (ODMs) with products, allowing them to build mobile phones for GSM/GRPS, EDGE, and WCDMA standards [3]. EMP's products consist of a hardware reference design and complete software stacks up to an application API, where EMP's customers can place their own applications. EMP's entire range of products is based on a small number of embedded platforms [2, 7].

In this context of embedded platform design, software performance is crucial due to a variety of reasons like the existence of real time critical tasks, advanced multimedia functionality as well as cost and energy constraints, to name some. Thus, work on software performance is carried out in many parts of EMP, representing various needs as well as providing various expertises. Amongst others, system designers, programmers, or product managers are involved in EMP software performance engineering (SPE) work. Typically, each of these groups has their specific requirements on SPE data. It is not trivial to use performance data [12] effectively over the entire company, even more so as the time needed to collect, organize and retrieve data for a complex system is not negligible.

On this background, a case study was carried out at EMP. The approach was to study which performance data is available or needed, to which purposes performance data is currently used, and which new usages are made possible by organising and structuring the available performance data. Based on this case study, a software performance database was proposed to organise and use software performance data effectively at EMP. The database was then implemented and evaluated.

2 Research Methodology

The main research question pursued is the following:

- How should SPE data be organised and made available to satisfy the needs and possibilities for the development of an embedded platform?

Three activities have been conducted to answer the research question. The first activity was an archive analysis of development documents to learn which data is available and who the users are. About 20 design documents were analysed during this activity. The second activity was a set of

interviews with selected personnel from the case organisation. The goal was to understand how the actual data is used and what types of performance estimations are of interest. The sampling of subjects was based on their knowledge and experience in a way that resulted in a broad spectrum of knowledge of the work in the organisation. In total 10 experts from various fields (system design, multimedia, audio, system test, digital signal processing, Java) were interviewed. The third and final activity was to propose, implement and use a method of organising performance data and estimations to support effective SPE.

3 Results from the archive analysis and interviews

Although a software performance engineering process is presented in [13] the result let us believe that a development of an embedded platform poses requirements in the development processes not covered by [13]. The overall results are covered in this paper, while others of a more detailed nature are of a confidential nature for EMP, and thus cannot be published.

The archive analysis revealed that available data is highly heterogeneous, and that it is spread over entire EMP. The heterogeneity reflects the different needs within EMP (e.g. system design vs. software optimisation) as well as the different methods to obtain data (e.g. estimations vs. measurements) also implying that the data is of different quality, as e.g. measurements typically provide more reliable data than do estimations. That SPE data is spread over the entire organisation is a consequence of the fact the SPE is an integral part of the entire embedded platform development. Moreover, different users do need the SPE data for considerably different purposes, related to their roles at EMP i.e. product managers, software developers, hardware developers or testers. The result can further be summarized in a number of requirements concerning the collection (Chapter 3.1), display (Chapter 3.2), usages and analysis (Chapter 3.3) of the software performance data.

3.1 SPE Data Collection

During the interviews and the archive study, it became clear that the concept of use case [11] is central. A use case represents something a mobile phone user accomplishes using the phone. Use cases play a central role as organising entity during specification, design, implementation, and testing. It should also be possible to represent scenarios of use cases, e.g. a sequence of parallel use cases, as this is of great practical importance. In the view of a hierarchical organisation, the scenarios would constitute the level above the use case.

Performance measurements should be associated with the use cases. These measurements should be representing at the hierarchy level below the use cases, further on referred to as tasks. The heterogeneity of the data makes it necessary that different types of measurements can be represented (e.g. workloads, call dependencies, or process timing distributions). Moreover, it mandates the storage of data origin. Since the data may originate from varying sources like for example, early estimations present in design documents or measurements made on different hardware or software releases.

It should be possible to add other relevant performance measures without altering the structure of data representation even when this data may not directly be associated to software performance, (e.g. memory bandwidth requirements, power consumption). To achieve these goals, the information about the hardware architecture must be included, as the SPE data is always tied to the hardware where the measurement was made. In the hierarchical structure of the SPE data, these hardware representations constitute the lowest level.

3.2 Data display and performance analysis

Concerning data display, it turned out, that a main requirement was the ability to quickly retrieve performance data on the different hierarchy levels (i.e. scenarios, use cases or tasks), to easily add more detailed analyses to the data display (e.g. process timing distributions instead of only focusing on workload). Moreover, producing customized graphs and specific statistical evaluations based on the stored data was perceived as a very valuable feature.

There should also be means to judge whether certain performance criteria are met, e.g. that the sum of the workloads for a scenario does not exceed the CPU capacity. Mcps (Millions of processor cycles per second) is used throughout the case study as the measurement unit for workload. Concerning the hardware representations, one requirement was to provide a way for defining operators for transformation of measurements between different hardware architectures. Considering performance estimations based on SPE data, it was considered as important, that estimation models could be easily integrated with the organised SPE data.

3.3 Usages of Organised SPE Data

The usage is listed for each development phase; Design phase, Implementation phase, Maintenance phase, Platform update.

(1) Design phase: During its initial stage, when decisions upon the system design of the platform are taken, one of the focuses is to set requirements on the hardware and software. The requirements are used as input to the next project

stage, when a more detailed design is to be developed. Considering different design alternatives, the predicted performance of future applications is an important factor. Therefore, the choice between different alternative designs is often based on their estimated performance. Typical design alternatives that constitute the design space during system design are e.g. concerned with hardware choices as: different CPU clock frequency and architecture (i.e. ARM CPU vs. Digital Signal Processor (DSP)), hardware accelerators, communication links, hardware/software partitioning, etc. Here it is seen as crucial to use existing performance data to estimate system performance for a new system design. For an efficient SPE process, this data needs to be well organised, easily retrievable and well integrated with estimation models and methods. A further requirement was to support automated analyses, so that the impact of changes in system design can be evaluated quickly. Also, the possibility to assess performance characteristics of the entire system (e.g. by making it simple to roughly estimate the performance implication of certain critical use cases) might be appealing to product managers as an easy to use mean of reviewing customer requirements.

(2) Implementation phase: During implementation, when the system design has been set, performance engineering can be used to find bottlenecks of the proposed design, by evaluating different design alternatives, using simulations or prototypes. For these tasks, it was seen as very beneficial if the performance data could be collected, organised, analysed in a standardized and automated way. Once bottlenecks are identified, the same SPE process can be used to track the results of software or system optimisation.

(3) Maintenance phase: In this phase well organised SPE data and analysis tools can be a means for tracking the effects of ongoing smaller design changes in terms of performance (i.e. change the partition of functionality on different hardware or software entities). The tool can also be used for assessing the performance implications of new usages (new scenarios and use cases) of the platform. Moreover, well-organised SPE data and analysis tools will be beneficial for system testing.

(4) Platform update, major configuration changes: When making major changes in the configuration of the platform, there is a need to know how different configurations impact on system performance. The configuration changes include both the software and hardware of the system. The system performance can be defined as the response times and workload for different applications. For example, knowing the threshold of the maximum workload allowed to achieve a desirable responsiveness can create design guidelines that the CPU must not exceed this specific workload. In case the platform must be updated in order to cater for new functionality and better system performance, there is a need to know how to re-design the platform in order to meet the desired

goals. Performance estimates needed for this purpose can be viewed in terms of output and input information to the development life cycle of an embedded platform where software and hardware co-design is a vital driving force. The possibility to use existing performance data to estimate the effects of the changes or updates will be very much facilitated through standardized, accessible, and well-organised performance data. A further benefit of well-organised, accessible and easy to use performance data, analysis tools, and estimation methods, is that the need for software engineers to get involved in the details of the hardware in order to make estimations of the software performance decreases. This is a time consuming operation for most software developers. Hardware simulators must be set up and the software application executed on it. Still, even with these efforts, the result will suffer in accuracy and certainty. This is due to the limited accuracy of the hardware model used in the simulations, which stem from the fact that when the estimations are needed there is no hardware to run the software on. The estimation is in fact used to take strategic decision about the design of hardware.

The conclusion is that well organised performance data, which is tightly integrated with analysis tools and estimation methods, is beneficial for all stages of embedded product development.

4 Proposal for organising data - a software performance database

In order to cope with the overall data heterogeneity and the distribution of data over entire EMP a software performance database is proposed for supporting EMP's SPE effort. The design of the database should fulfil the presented requirements on data, analysis and estimation tools. A database allows improving effectiveness of SPE by helping to define, organize, and communicate the company performance strategy; and to translate strategy into operational objectives related to performance work. With the continuous loop of insight into the estimates and measurements that can be provided by the database, the company can easily measure effectiveness and continually refine the SPE methodology by for example continuously validate the estimations against a real product.

There exist alternatives to collecting the SPE data into a central database, which after all takes resources to build up and administrate. It is however believed that a database decrease the situations where each group involved in a software performance work, independently collects, defines, and uses its own software performance data whenever needed.

4.1 Database requirements

The requirements elicited from EMP and discussed in Chapter 3 can be rewritten in the context of using a database solution. The rewriting gives rise to a requirement list (DB1-DB7).

DB1. The database should be a central repository for all performance data in the company.

DB2. The database should allow for easily retrieving and visualizing performance data.

DB3. The database should be able to represent any type of performance data (e.g. be processor load, memory bandwidth, or power consumption).

DB4. The database should allow for aggregation of entries (e.g. summing up all data on OS tasks for one use case)

DB5. The database should be a tool for a wide range of prediction activities in all development phases (e.g. estimation of CPU load of sets of use case, calculation of response times and latencies)

DB6. The database should allow for integration with a wide range of evaluation and analysis tools for software and system performance estimation. (e.g. static models, dynamic models, simulation kernels).

DB7. The database should allow for visualization of analysis results on different levels, e.g. on use case level, RTOS process levels, or algorithm levels.

4.2 Design of the Database

In the context of the development of EMP's next platform architecture, a first design of a performance database has been specified and implemented. The first usage objective at hand was to collect and organise performance data of the existing platform architecture, and use it to estimate the performance of the new platform performance.

For this first implementation of the database, it was decided to use the workload (Mcps) consumed by specific tasks as primary performance measure. Workload is a very important performance measure, and much of EMP's performance data is available in terms of workload per task. It was also decided to organise the performance data in terms of (parallel) use cases. This because a goal with the performance estimation was to judge whether the use cases the platform is designed for can be realized with the proposed system architecture.

A strong focus of the design has been to use the data existing at EMP and to have means for easily integrating

them into the database. Existing data can be found in current platform design proposals, use case validation, in measurements from the software groups as well as concurrency measurements conducted by the system test group. In a next step, the database is to serve as input for more sophisticated performance estimations based on queuing models and simulations. The data analysis was conceived with this aim in mind allowing the extraction of data on e.g. timing distributions or call matrices on RTOS process level.

4.3 Structure of the data representation

The structure of the database is hierarchical and heterogeneous see Figure 1. The top-level entities are scenarios

Figure 1. Data representation in the database.

reflecting sequences of parallel use cases. A typical scenario would e.g. be to (1) turn on the phone (2) download of file over GPRS and at the same time receive a voice call. The entities scenarios consist of are use cases. As most of the customer requirements and internal design requirements are formulated in terms of the possibility to execute specific use cases on the platform, use cases can be seen as the central entity of database organisation. For the initial version of the database, the use case table was directly extracted from the "use case requirements document", listing and describing the use cases the current platform has to handle. For means of representing performance figures, each use case is composed of a number of tasks. A task relates to something that can execute on a device's hardware, e.g. an algorithm, an RTOS process, a group of RTOS processes, or simply a number of CPU cycles. The use case "GPRS file download" e.g. consists, amongst others of, a "datacom task", reflecting all processes related to the datacom stack, and "GSM access" task, related to the GSM access processes.

Which hardware is involved and which resources are consumed is held in the hardware descriptions, which constitute the lowest level of the database. Each task may contain several hardware descriptions reflecting the task executing on different hardware platforms. The hardware descriptions are flexible structures, reflecting the need to represent different hardware architectures and measurement figures as well as the necessity to abstract the software processes from the hardware they execute on. In the current version of the database two hardware descriptions are defined, one for the current platform and one for a possible upcoming platform. Currently, the tasks only contain values on the workload, but might also involve other hardware resources like memory bandwidth or power consumption. Consumed hardware resources per task during low CPU load and the maximum available workload for the respective hardware differentiate these values. The latter value can be seen as the workload budget for that specific hardware. An example of the data represented in the hardware description for a released platform is showed in Table 1, representing the datacom task of the use case mentioned earlier. The 'budget' column reflects the maximum Mcps available for the respective hardware. The 'task requirement' column represents the number of Mcps used by the specific task. The values used are fictive.

Table 1. Data represented for the Task "Datacom" in the hardware description for the current platform.

Current Hw, task Datacom	Budget	Task requirement
ARM Mcps	100	10
DSP Mcps	100	2

4.4 Transformations to new hardware architectures and queuing models

The transformation of performance data measured on one hardware set up is transformed to different hardware architectures by means of scaling factors, e.g. assuming that a task consuming 100 DSP cycles will consume 300 cycles on an ARM CPU. Obviously, very detailed knowledge of the transformed task and the hardware set-up is needed to get an accurate estimation of the multiplication factors. As this knowledge may not be available in the first phases of the development cycle, these factors introduce an uncertainty into the generated performance estimations. In the current implementation, no explicit data is available of how the tasks make use of caches, nor is bandwidth of data- and instructions buses available. This is not a restriction when performance estimations are calculated for the same hardware as the measurements are made on, since these factors

are implicit to the measured performance data. This is however not true for performance estimations obtained for new hardware architecture where these factors can imply large uncertainties because not considering use of caches, nor the bandwidth of data- and instructions buses. In addition, to capture the dynamic behaviour the transformed measures are used as input to queuing models such as the layered queuing network (LQN) model. A layered queuing network (LQN) is an extension of the queuing network models [10]. LQN defines a system in terms of request sent and service given by different hardware and software entities. The entities can be divided in three difference categories: client tasks that only requests service, client-server tasks that can both receive and send requests and server tasks that only receives requests. These entities are placed in different layers where an entity at a higher level where an entity from a higher level is allowed to request service from a lower layer, but not vice versa. Modelling a system as an LQN provides analytical means to estimate the system performance with different software and hardware configurations with different workloads.

5 Usages for EMP

5.1 Data collection

The database can be used as a central point of collection for the regularly conducted performance measurements by the various design teams. As the database is accessible from any workstation, performance figures will become more easily available even outside the respective design teams. This greatly increases the possibilities make performance testing and monitoring an integral part of system design.

5.2 Data Retrieval and visualization

On the most elementary level, the database can be used to retrieve the performance figures stored in various ways: either by browsing scenarios, use cases or tasks, alternatively by searching for a specific use case or task. In case scenarios or use cases are concerned, the current version of the database will report the sum of Mcps of all tasks associated with the respective hardware entity and thus give a figure for system load. Moreover, for the data obtained from trace files, detailed per process statistics are available in addition to the task figures. The statistics currently implemented are relative execution times per process for a use case and, a set of time distributions for all processes per measured use case as shown in Figure 2.

In time for Process VIE_ENCODERP

Figure 2. Timing distribution for a process.

5.3 Performance estimations

The database can be used to make performance estimations based on existing measurements and estimations. One example is to estimate performance figures for parallel use cases based on the measurement of individual use cases. This is especially interesting for system design issues, as the database collects information about the entire system (e.g. DSP and Application CPU). Thus the group responsible for development of the DSP gets the possibility to estimate the effects of their performance work on the entire system, something which turns out to be difficult if data and estimation methods are not available through one single interface but rather are spread out over the entire company. Another example is to use the data available for the current platform to do performance estimations for an upcoming design. By specifying the transformers for the tasks affected by the changes in the hardware platform, rough performance estimations for the new hardware configuration can easily be obtained. In our implementation, a transformer distributes the performance figures measured on the existing design's different hardware components (DSP and ARM) to the new design's processors (DSP' and ARM') applying simple scaling factors. Once again, the database as a central collection point makes such estimations much more feasible, as all data is readily available and the estimation models can be integrated into the database. If the algorithms or methods are altered and used for specific software tasks, the figures in the database and the performance esti-

mation can be quickly updated as to always reflect the most up to date system parameters. Although the above described performance estimation methods have their limitations (see Chapter 6), there is today no alternative way for quickly getting a large number on estimations based on measurements. If these estimations are treated as indications and a means to identify possible performance bottlenecks, instead of considering them as exact numbers, they can be very beneficial for the system design process.

5.4 Example - CPU Budget

During the design, work with a platform upgrade the performance database was used to calculate a "CPU budget" for the upgraded platform. The platform upgrade implied that parts of the software system were to be redistributed from CPU1 and CPU2 to two CPUs of other type CPU1' and CPU2'. The first step in this work was to identify the most critical use cases amongst others videoconferencing and MP3 playback. For these use cases a set of measurements were conducted on the existing platform and imported into the performance database by means of the integrated trace-file analyzer. By means of these measurements and the architecture description of the upgraded platform (memory bandwidth, cache system and CPU characteristics) the task transformers were determined. Moreover, two sets of alternative hardware descriptions for the upcoming platform were integrated in the database. After these activities were completed the database was used to produce performance estimations for the alternative hardware descriptions of the platform which directly showed the CPU budget (Table 2). CPU1 and CPU2 represent measured MCPS for the existing platform while CPU1' and CPU2' represent estimations for the upgraded platform. The actual CPU budget consisted of the sums of all tasks (not reported here) for the different CPUs.

Table 2. Data in Mcps for three tasks in the videoconferencing CPU budget.

Task	Current		Upgraded	
	CPU	CPU2	CPU1'	CPU2'
PHY	0	0.8	0.9	0.2
BSS	0	1.8	2.0	0.4
DEC	2.5	0	0	10.5

6 Restrictions of the proposal

When using the database some reservations must be made. They arise from restrictions of the measurements, the

representation of hardware and software architecture and finally by methodologies used for obtaining performance estimations.

6.1 Non independent use cases

Typically, the use cases are non-independent in terms of software that is executed. Two different use cases may share the same software resources. This is not considered in the currently implemented model, which simply adds up processor cycles for two separately conducted measurements on two use cases. This can lead to that data from OS processes are being included twice in the resulting performance estimations. The resulting figures are thus an overestimation of the required processor cycles.

One way to handle this problem is to detect the common resources and to consider this when calculating performance estimations. Another way to attack the problem is to assume (or better investigate) that shared service times are negligible in relation to overall service times and explicitly state the resulting estimations as upper bound estimations.

6.2 Hardware transformations

Although the performance estimation mechanisms in terms of hardware transformers implemented in the database are plausible and straightforward, the lack of validation implies that performance estimations have to be seen as rough estimates rather than facts.

7 User Experiences

This chapter reports these collected experiences from all these activities together with the authors' own experiences of using the database tool. They cover different aspects in line with the research questions, as having a centralized repository for storage of measurement data, retrieval and visualization of data, as well as performing analyses and estimations.

7.1 Central repository

It is evident that different users can use software performance data in many different ways and therefore might have different expectations on the database. Nevertheless, all users were positive of using a central repository. This seems obvious at first glance but there is to remember that using the database initially imposes an additional effort and new routines to the users. The initial effort is in comparison of using an own local internal representation and storage. When asked a direct question about the benefits of the database, users believe that the benefits gained are greater than the extra effort required by the database. So, when the

total experiences from the users point to the fact that that all are willing to put up with the extra time and effort when using the database, this is considered a very positive sign to the usefulness of the central repository structure of the database.

7.2 Storage of measurement data

The main sources of measurement data in an embedded environment are analysis of trace-files for RTOS processes. Other measurements done in the development are for example the processor cycle counts for algorithms in a digital signal processor (DSP). Both these types of data did fit very well into the structure and usages of the proposed database. The flexibility and rather small scale approach has allowed to quickly implement new features and analysis methods, as well as to change the database organisation to support the demand of the its potential users easily. In this way it is believed that it has been possible to some extent tailor the database to an embedded platform development organisation such as EMP. During the development of the database, additional suggestions of requirements have been collected. Two examples are the following:

- To present the optimization grade of the software process or algorithm. The optimization grade would be used to know if there are possibilities of reducing the CPU load or achieve a faster response times for applications, by optimizing single software tasks.
- The hierarchical decomposition of the use cases has been perceived as useful and natural to use. However, there have been requests of other grouping, more related to the structure of software development organisation.

7.3 Retrieval and visualisation of information

The user interface provided was appreciated the users. The appreciation was much based on the hierarchical structure implemented, which facilitated the quick access to the data on the level most relevant to the respective users. It did support all interested parties of software performance in exchanging, collecting and estimating software performance. The support was present on both the task view; use case view and scenario view. Support was also given to features like call dependency graphs and statistical data analyses.

7.4 Software performance estimations

The database has been used as the main source for retrieving data for different estimations models during a project. The decision of using a hardware abstraction layer has been experienced as very positive. Even for experts in

software/hardware co-design and co-simulations it can be stated that it takes a non-negligible effort for producing the estimations needed. This is the rationale behind the proposal of the implemented and used hardware abstraction layer. The implementation of the hardware abstraction layers by using simple multiplication factors to separate the execution time between different hardware has been well accepted by the individuals involved in SPE. The acceptance has not only been from software developers, but also by the experts that have delivered the factors. The reason is the possibility of continuously during the development adjusting the multiplication factors, and observing the impact, the estimated factors have on the software performance (estimations) at system level. The possibility of separating the system software estimation from the hardware model estimation is yet another experience that supports the proposed database organisation and usage. This is a fact that becomes even more obvious when a new platform is to be developed, where performance estimations are done considering a baseline (i.e. the current platform which can be executed and profiled). The multiplications factors are then estimated from the old platform to the new platform, supporting the system estimation of the new hardware. A negative experience was that there was not enough support for identifying tasks that will not be executed twice when adding together separate or the same scenarios (i.e. the estimation of parallel scenarios). This restriction has been discussed in Chapter 6.1.

8 Conclusion and Future work

The first round of interviews showed an inhomogeneous requirement picture reflecting the different focuses of various groups involved in embedded platform development. When presenting the idea of the database a lot of expectations were raised from these different groups. The database has met the majority of the expectations and it is therefore valuable as base for the software performance work which is requested from within EMP. The structure and usages has also facilitated the communication between different parties involved in the software performance discussions. The conclusion is that the proposed performance database is a valuable tool for embedded platform development. The hierarchically structured database, organised at the highest level in terms of scenarios of parallel use cases and the tight integration with estimation models is perceived as an appropriate solution to the requirements discussed in Chapter 3. Moreover, the proposal was considered as beneficial for most phases of embedded platform development.

In the future more analysis methods will be implemented, such as for example RMA calculations [9] for scheduling purposes, queuing and simulations models for estimation of responsiveness. Furthermore, the database should extend the support of parallel scenarios, by for exam-

ple extracting the common and shared software processes used by different scenarios. There might also be a need of more advanced hardware models used in the hardware abstractions models if needed, if it is discovered that the ones used does not fulfil all the needs of SPE work. Although good experiences have been received, it is important to realize that a more extended validation is needed for the parts of the proposal concerned with performance estimations.

9 Acknowledgment

The work in this paper has partially been financed by the Swedish Agency for Innovation Systems project number P23918-2A. Thanks and appreciation to the employees from Ericsson Mobile Platforms AB who participated in the study.

References

- [1] A. Bechini and C. A. Prete. Performance-steered design of software architectures for embedded multicore systems. *Software Practice and Experience*, 32(12):1155–1173, Oct. 2002.
- [2] P. C. Clements and L. Northrop. *Software Product Lines: Practices and Patterns*. SEI Series in Software Engineering. Addison-Wesley, Aug. 2001.
- [3] D. Collins and C. Smith. *3G Wireless Networks*. McGraw-Hill, 2002.
- [4] <http://www.ericsson.com/mobileplatforms>, 2004.
- [5] E. Gelenbe, editor. *system Performance Evaluation: Methodologies and Applications*. CRC press, 1999.
- [6] P. J. King. *Computer and Communication System Performance Modelling*. Prentice-Hall, New York, 1990.
- [7] M. H. Meyer and A. P. Lehnerd. *The Power of Product Platforms: Building Value and Cost Leadership*. Free Press, New York, 1997.
- [8] G. D. Micheli and R. K. Gupta. Hardware/software co-design. *Proceedings of the IEEE*, 85(3):349–365, March 2002.
- [9] R. Nord and B. Cheng. Using rma for evaluating design decisions. In *Proceedings of the IEEE Workshop on Real-Time Applications*, pages 76–80, 1994.
- [10] J. A. Rolia and K. C. Sevcik. The method of layers. *IEEE Transactions on Software Engineering*, 21(8):689–700, Aug. 1995.
- [11] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual*. Addison-Wesley Publishing Company, 1998.
- [12] A. Smith. The need for measured data in computer system performance analysis or garbage in, garbage out. In *Proceedings of the 18th Computer Software and Applications Conference*, pages 126–431, 1994.
- [13] C. U. Smith and L. G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison Wesley, 2002.
- [14] W. Wolf. Hardware-software codesign of embedded systems. *Proceedings of the IEEE*, 82(7):967–989, July 1994.