Summary

An embedded system is a software-intensive system that has a dedicated function within a larger system. Embedded systems range from small portable devices, such as digital watches and MP3 players, to large complex systems, like road vehicles and medical equipment. Embedded systems have increased significantly in complexity over time and are confronted with stringent cost constraints. They are frequently used to perform safety critical tasks and their malfunctioning can lead to serious injury or even fatalities, as with medical systems.

Embedded systems interact with their environments in a time-critical way. Hence, their *safety* is predominantly determined by their performance. Performance is the amount of work a system accomplishes over time, which is frequently expressed in, among others, terms of response times, throughputs, utilizations and queue sizes. Good performance is hard to achieve, because: (i) performance evaluation is hardly ever an integral part of software engineering; (ii) system architectures are increasingly heterogeneous, parallel and distributed; (iii) systems are often designed for many product families and different configurations; and, (iv) measurements that gain insight in system performance tend to be expensive to obtain.

In this thesis, we consider so-called *service(-oriented) systems*, a special class of embedded systems. A service system provides services to its environment, which are accessible via so-called service requests, and generates exactly one service response to each request. In a service system, service requests are functionally isolated, but can affect each other's performance negatively due to competition for shared resources. Evaluating the performance of service systems before they are fully implemented is hard; answering performance questions in this domain calls for models in which timing aspects and mechanisms to deal with uncertainty are combined. Examples of uncertainty are the use of statistical predictions for execution times, two parallel processes competing to use a resource first, and system parts that are only partially specified in the model.

We propose a new performance evaluation framework for service systems. The system designer models the performance of a system, leading to a *high-level performance model*. Under the hood, this model is transformed into an *underlying performance model*, which is *evaluated* for performance. This leads to *performance results*, which are, in turn, presented to the system designer.

The literature reports about many so-called toolsets that facilitate the performance evaluation process for the system designer. A toolset is a collection of connected tools that automates (part of) the performance evaluation process. The majority of the toolsets, however, require the user to have knowledge of formal performance evaluation techniques. A system designer normally does not have this knowledge, because these techniques are not part of his daily practice. On top of that, many toolsets provide a high-level model that is *not* domain specific. This leads to a large modeling effort, and hinders the understanding and communicability of the model. Morever, most toolsets tend to return approximate results, because the generation of exact results generally relies on exhaustive evaluations. Finally, it is not uncommon that the system designer needs to perform labor-intensive and error-prone efforts to aggregate and visualize results.

The goal of this thesis is to overcome the above shortcomings; the system designer should be able to create a performance model in a language of his domain and obtain results in terms of this domain. This calls for a *high-level model* that is domain specific, expressive and concise. The model should also explicitly separate the application (software) from the platform (hardware) to support the independent evaluation of combinations of them. Also, the model should facilitate calibrations based on real measurements to make the model more realistic. Moreover, *performance results* should be presented in a visual, comprehensive way and cover a variety of performance metrics, e.g., utilizations, queue sizes, throughputs and latencies, to gain a deep insight in the underlying system's performance characteristics. Finally, the system designer should be able to perform *fully automated evaluations*, for many designs and modes of analysis, and within a practical amount of time.

To achieve the goal of this thesis, we have designed and implemented iDSL, a language and toolchain for performance evaluation of service systems. Interaction with system designers conveyed that they find the iDSL language intuitive, presumably because of their familiarity with the service systems domain. Moreover, iDSL models were perceived to be remarkably concise and at the right level of abstraction. Also, the iDSL language suppports the separation of application and platform, as well as calibrations. System designers appreciated the level of automation of the iDSL toolchain; after providing an iDSL model as input, visualized performance results for many designs are automatically returned. Under the hood, the so-called Modest language and toolchain are used to evaluate models using advanced model checking algorithms and discrete-event simulations. Modest has been selected because of its process algebra roots, its relatively readable language, and its support for multiple formalisms and ways of analysis.

For accurate results, iDSL uses a new evaluation technique based on probabilistic model checking, which yields full latency distributions. For this purpose, iDSL uses an algorithm on top of Modest, in which many iterations of Modest evaluations are performed in a systematic way. The evaluation technique exhaustively traverses the model, making the evaluation slow. iDSL counteracts this with automated model simplifications and making use of light-weight evaluation techniques initially that speed up the evaluation.

The applicability of iDSL on real systems has been assessed by conducting several case studies on *interventional X-ray systems* throughout the development of iDSL. Interventional X-ray systems are dependable medical systems that support minimally-invasive surgeries. Additionally, the wider applicability of iDSL has been assessed by applying iDSL on a load balancer case study.

In conclusion, iDSL delivers a fully automated performance evaluation chain from a high-level model to visualized performance results for service systems. This approach is not only very efficient, it also brings advanced formal performance evaluation techniques at the fingertips of system designers, without bothering them with the technical details of these.