Uniting Academic Achievements on Performance Analysis with Industrial Needs*

Bart Theelen[1] and Jozef Hooman[1,2]

[1] Embedded Systems Innovation by TNO, Eindhoven, The Netherlands
[2] Radboud University, Nijmegen, The Netherlands

Abstract. In our mission to advance innovation by industrial adoption of academic results, we perform many projects with high-tech industries. Favoring formal methods, we observe a gap between industrial needs in performance modeling and the analysis capabilities of formal methods for this goal. After clarifying this gap, we highlight some relevant deficiencies for state-of-the-art quantitative analysis techniques (focusing on model checking and simulation). As an ingredient to bridging the gap, we propose to unite domain-specific industrial contexts with academic performance approaches through Domain Specific Languages (DSLs). We illustrate our vision with examples from different high-tech industries and discuss lessons learned from the migration process of adopting it.

Keywords: Performance Modeling, Performance Analysis, Quantitative Analysis, Domain Specific Languages, Model Checking, Simulation

1 Introduction

Quantitative qualities of high-tech (embedded or cyber-physical) systems are often key selling factors second to the ability to perform certain functionality. As opposed to functionality, quantitative qualities can often be balanced against each other, possibly across multiple technologies or engineering disciplines. One may for example realize a better performance by either changing the software algorithms that implement the functionality, by changing the configuration of the resources executing the algorithms or by some combination of both, where the trade-off can have an effect on other quantitative qualities such as physical size of the system. The diversity in possible design trade-offs that can be made for quantitative qualities has resulted in a plethora of academic results on how to design high-tech systems effectively and efficiently. Despite these achievements, which are commonly based on the use of abstract models to predict the qualities of a design proposal or even construct an appropriate design, high-tech industry still relies mostly on traditional approaches leading to a reactive way of working in resolving unexpected issues during integration and test or even later.

* This work was supported by the ARTEMIS Joint Undertaking through the Crystal project on Critical System Engineering Acceleration and the Dutch program COMMIT through the Allegio project on Composable Embedded Systems for Healthcare.

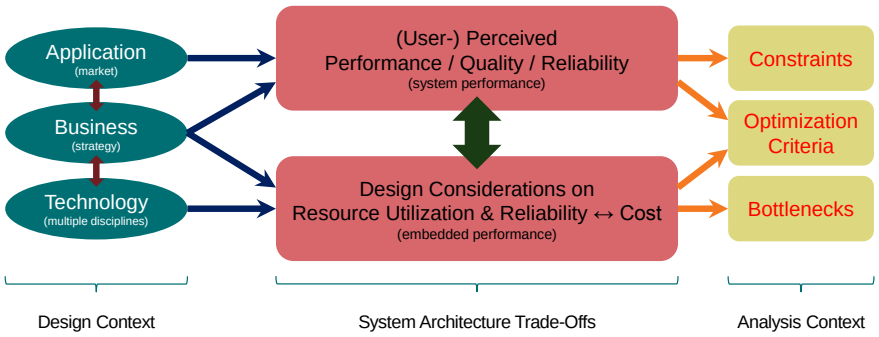

Fig. 1. Performance in an industrial context.

Figure 1 illustrates that achieving appropriate quantitative qualities during industrial design processes requires system architects to make trade-offs subject to a complex design context. This design context is characterized by the application functionality demanded by (multiple) markets and the business strategy towards serving the diverse and sometimes conflicting market demands. The design context also covers combining technologies from multiple engineering disciplines that may be subject to business strategy-depend restrictions on availability or that may be developed as part of the business' innovation processes. A related aspect is that most designs in industry increment from a long-standing legacy for which the rationale behind certain trade-offs may no longer be known or valid.

In trading off design alternatives, the center part of Figure 1 expresses that system architects have to consider the impact on the user-perceived system qualities (i.e., the qualities listed in a product catalogue for which users pay) and the qualities that are relevant from a business and design perspective (like cost and physical size). Examples of user-perceived qualities include the throughput of a printer system in terms of pages printed per minute or the quality of a picture on a TV display. Notice that such qualities are sometimes difficult to express in measurable objective metrics. An important difficulty to deal with is the unclear or complex relation between user-perceived system qualities and the measurable objective metrics used for evaluating design alternatives. How does the load of a processor for example impact the quality of the picture on the TV display?

Evaluating alternative designs involves exploiting analysis techniques for different kinds of quantitative qualities. The right-hand side of Fig. 1 categorizes qualities as constraints, optimization criteria or bottlenecks. The difference between constraints and optimization criteria is visible from whether the requirement specification includes a concrete target quantitative value that must be satisfied. For example, the printer throughput must be at least 80 pages per minute (constraint) at a minimal cost (optimization criterion). Bottlenecks relate to design considerations causing a difficulty in satisfying constraints under specific, often unforeseen, usage conditions. Bottlenecks often arise due to legacy.

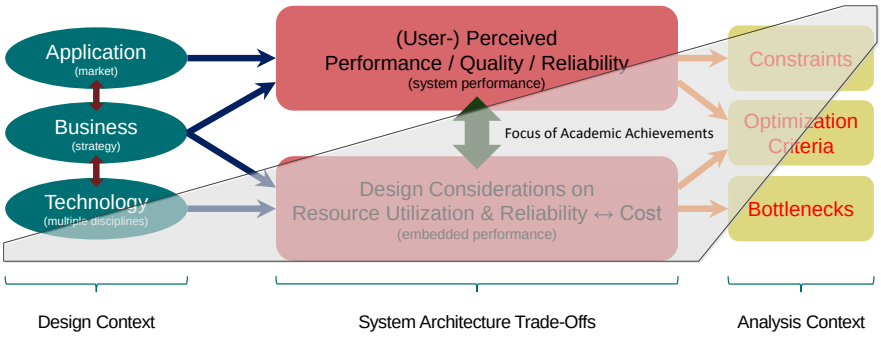

Fig. 2. Focus of academic achievements in relation to industrial needs.

In this paper, we investigate the difficulty of getting academic approaches adopted by industry. We start with highlighting a mismatch in focus of the academic achievements in relation to the industrial needs as shown in Fig. 2.

It is natural for academic work to disregard most business and market aspects. Trade-off analysis is commonly limited to a single engineering discipline or just to the technologies of embedded systems (i.e., computer electronics and software). The industrial practice may however also cover for instance mechanical or optical technologies combined with choices for different materials. A well-known example from the past is the CD-player tray where the original choice was for expensive stiff materials to give support to CDs in order to realize accurate reading by laser (requiring simple control software) as opposed to the cheaper plastic trace developed later which is too flexible from a mechanical viewpoint and therefore requires more complex control software to read the CD accurately.

Due to the diversity of applications, academic approaches rarely cover the relation between embedded performance and system performance unless the relation is reasonably straightforward. Although for example the quality of a picture on a TV display and the accuracy of overlaying ink by a printer may depend on the same embedded performance metrics, the relation between them is very different and therefore hampers the typical generalization pursued by academia.

With respect to the analysis context, academia concentrate on constraints and optimization criteria. A mismatch we observe here is that industry often only cares about getting approximate results for a large variety of quantitative qualities very fast whereas academia promote the use of formal methods because they give high quality results, while suffering from limited scalability. An additional issue with formal methods is that they are commonly perceived as not being very intuitive. Moreover, industry struggles mostly with identifying bottlenecks. This shows from the need to evaluate the concrete value for a wide range of quantitative qualities under certain conditions instead of/next to pursuing satisfaction of a certain bound or an optimization. Many academic approaches (in particular formal methods) do not support such analysis (straightforwardly).

A primary difficulty is in fact the question of what approach or tool to apply for which purpose. Academic approaches claim their capabilities on small case studies with assumptions that rarely hold in practice. As a consequence, developing, calibrating and validating performance models at a suitable abstraction level to make academic approaches work often turns out to be very difficult. In this paper, we propose to unite (multiple) academic approaches with industrial needs by means of domain specific languages (DSLs) [29]. This allows providing an intuitive front-end to industrial users, while exploiting the rigidness of academic approaches for their envisioned purposes in any specific industrial context.

The remainder of this paper is organized as follows. The next section presents the main ingredients of our approach. Section 3 describes several lessons that we learned on getting our approach adopted by industry. Section 4 illustrates application of our approach in three different industrial domains. After reflecting on related work in Sect. 5, we summarize our conclusions in Sect. 6.

2 Approach

Industrial design processes are commonly *reactive* when it comes to performance qualities, i.e., performance is realized/optimized as an 'afterthought' when the design is already fixed. This is often due to the (business) needs of exploiting certain legacy or reducing costs while extending the functionality to support (market). As a consequence, performance is only considered when substantial bottlenecks arise during test and integration (i.e., the right-hand side of the V-model [8]) or even later. To counter this, we promote taking performance aspects into account right from the start of the design and system architecting phases (i.e., the left-hand side of the V-model). Academia proposed many predictive and some constructive approaches to support this vision. Following primarily an analysis strategy, *predictive* approaches exploit performance models to allow iterative evaluation of design alternatives devised by system architects before selecting the most appropriate one that is eventually realized by certain implementation technologies. A step further are *constructive* approaches, which follow an automated synthesis strategy in realizing 'performance-by-construction'.

As ingredient to introducing 'performance-by-construction' into industry, we propose the approach in Fig. 3. Crux is exploitation of a *reference architecture* or *architecture template* that describes what components a system may consist of and how these components can be combined into systems[3]. Such a reference architecture is available as design knowledge from the legacy in existing products or product families. Reconstructing the design knowledge and rationale on how components can be combined and configured *exactly* and how different concepts in the reference architecture interrelate is a very difficult process since this information is commonly ill-documented. We propose the use of DSLs to formalize this knowledge such that it can be processed automatically for the purposes of

[3] Notice that the terms *component* and *system* are relative to the level of detail at which the design takes place where the highest level denotes complete products and the lowest level comprises automatically synthesizable or bought off-the-shelf parts.

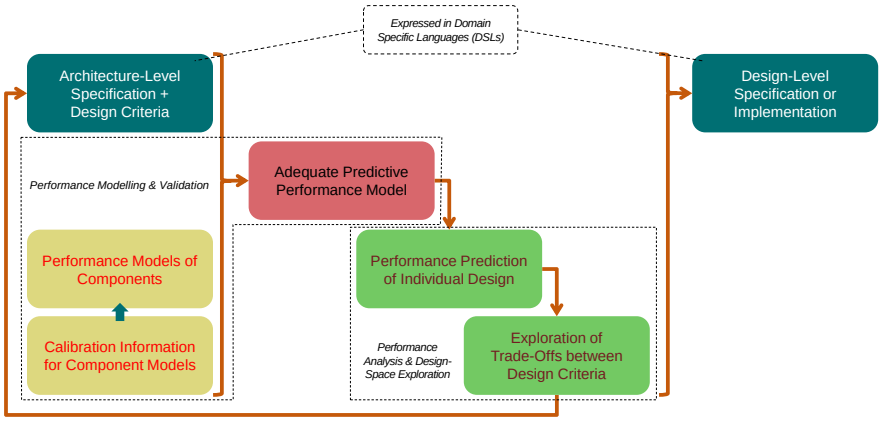

Fig. 3. An approach towards achieving 'performance-by-construction'.

both (system & embedded) performance analysis and synthesis. Exploiting DSLs allows specifying designs and any qualitative quantity of interest in a way that is close to the intuition of the system architects by using their domain specific terminology and way of describing things [16], while hiding the complicated details of applying the (formal) analysis techniques from them. We experienced that deducting the knowledge underlying a reference architecture by formalizing it in DSLs usually reveals already many ambiguities and inconsistencies in what different system architects (often of different departments) understand about their designs, particularly at the level of detail necessary to automate validation of such possible ambiguities and inconsistencies in design specifications and to automate performance analysis and synthesis from valid design specifications.

The approach of Fig. 3 exploits the architecture-level specification of a design in terms of a DSL instance of the reference architecture to automatically generate (an) adequate performance model(s) for the quantitative qualities of interest. This is enabled by a collection of appropriately calibrated performance models for the individual components that conform to the reference architecture. Calibration of such model components is commonly a very difficult aspect in industrial practice. Although techniques like static code analysis can aid obtaining the required calibration data, one often has to rely on (extrapolation of) measurements of existing products (if measurements are possible at all). Nevertheless, isolating the contribution of individual components to specific measurement results, which is often needed for appropriate calibration, is in many cases infeasible and we observe a lack of academic approaches to improve on this.

Given the diversity of quantitative qualities an industry may be interested in, our approach supports the use of different analysis technologies. From the very same DSL instance, we may for example generate complementary analysis models ranging from basic formula-based computations to simulation models

and models for model checking tools depending, amongst others, on the metric of interest, desired accuracy and analysis speed. Notice that this requires appropriately calibrated component models for each of these analysis technologies and a proper understanding of the (behavioral) semantics underlying the components in the reference architecture. This (behavioral) semantics is not captured by the (static) semantics focused definition of DSLs, but it is captured as part of the generation algorithms. Since these are specific to each different target, a clear consistency challenge arises when developing design tools that realize Fig. 3.

Despite the ability to support complementary analysis techniques, we experienced that formal methods tend to lack applicability in terms of flexibility and scalability for addressing real-life industrial problems. We therefore often resort to simulation, which is in fact a commonly used approach by industry. However, the difference is that we promote the use of simulation tools that are based on formal methods and that provide some information on the accuracy of the results as far as possible (e.g., by using confidence intervals). Formal tools that we have successfully applied include POOSL [25], UPPAAL [5] and MODEST [10, 11].

Executing the appropriate analysis techniques for an individual design yields a performance prediction that serves as input for an overall trade-off analysis when comparing alternative designs. The analysis results for individual designs and the comparison of alternatives are again to be presented to system architects in an intuitive form. Hence, some post-processing is usually required in the design tooling to represent the raw quantitative results produced by an exploited analysis tool as they are not made available in the domain specific terminology. For this purpose, we created the TRACE visualization tool [13] that can be configured to show quantitative results conform the domain specific terminology.

When a design alternative is selected for realizing the system, the DSL instance of the corresponding architecture-level specification is used as starting point for generating a design-level specification DSL instance or synthesizing an implementation. If no suitable design alternative is identified, the exploration process iterates. Notice that such iteration may request for (re-)designing components for which calibrated performance models must be provided in order to complete the process. We therefore stimulate developing and calibrating performance models as an integral part of the design process (see also Sect. 3).

A final ingredient to the approach in Fig. 3 is to structure the DSLs conform the organization of an industry in terms of groups or departments. Hence, we have not one but a coherent collection of inter-dependent DSLs to capture a single design specification. The dependencies cover in fact the cross-department relations between their individual responsibilities. By the ability of DSL technology to automate validation of possible ambiguities and inconsistencies in design specifications, communication between system architects from different departments (often covering different engineering disciplines) improves substantially.

Starting from a performance modeling perspective, we have observed that the Y-chart modeling paradigm [14] shown in Fig. 4 fits very well to most industrial organizations. The Y-chart modeling paradigm follows the separation of concerns from [20] by isolating modeling the *application* functionality from

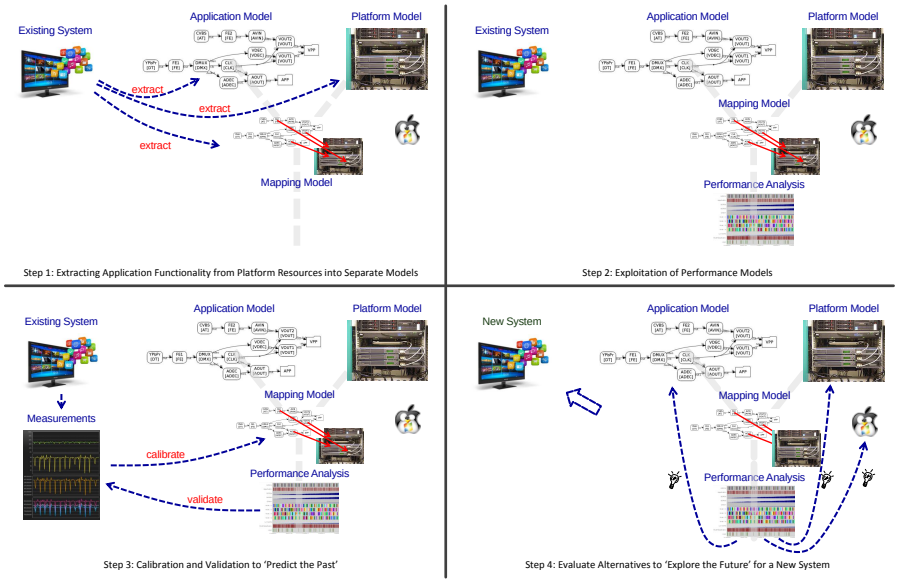

Fig. 4. Introducing the Y-chart modeling paradigm into industry.

modeling the *platform* resources that execute this functionality. The platform model covers the raw resources (time, space, bandwidth and energy) and the scheduling and arbitration mechanisms used to allow sharing them by multiple parts of the application. The *mapping* part of the Y-chart modeling paradigm covers the deployment of application functionality onto platform resources and the corresponding configuration of the scheduling and arbitration mechanisms in the platform. The crux of the Y-chart is to ease design-space exploration without the need to change the complete design specification when evaluating alternative mappings or platform configurations or when changing the application functionality for product variants. Moreover, the ingredients of application, platform and mapping are often addressed in industry by different departments as we exemplify in Sect. 4, while system architects (often organized in yet another department) are responsible for the quantitative qualities of their combination.

3 Lessons Learned

Industrial adoption of model-based engineering in general and formal methods in particular does not happen over night. We experienced that such adoption entails a migration process covering a cultural change in the way of working (and possibly also an organizational change), which takes about 2–4 years. Figure 5 identifies the phases we observed in this migration process, which we will classify in this section onto 'performance-by-construction' maturity levels 0–5.

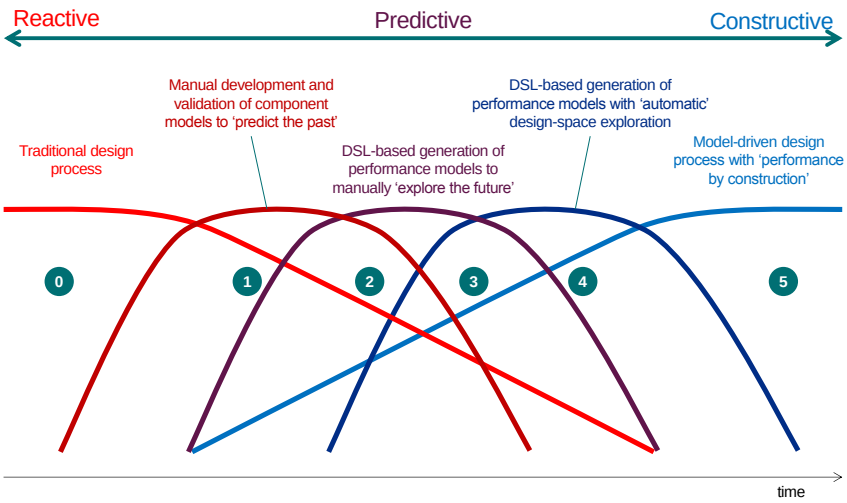

Fig. 5. Migration from a reactive to a predictive or even constructive design process.

Starting from a traditional design process (maturity level 0) where performance modeling and design-space exploration are processes during integration and test, the introduction of the Y-chart modeling paradigm in Fig. 4 provides some important added values to industry. The first is that the performance models extracted from an existing system are often a first-time formalization of design specifications, which thereby provides an aid in resolving inconsistencies in the design. The second (and possibly more important) value is the fact that the developed performance models give appropriate insight in existing bottlenecks as an aid to suggest better designs. A third added value comes from executing the last step in Fig. 4, which allows to predict the performance of any such design proposal. We classify this situation, where the exploration of design alternatives is performed based on manually constructed performance models (i.e., not generated from DSLs), as maturity level 1. The performance models are in this phase made by modeling experts who will however need to learn the industrial domain. To accelerate this and to allow the industry observing the added values, it is of utmost importance to perform real-life case studies without compromises and which are reasonably 'hot' in the sense that the observed issues may actually imply a loss in profit. These case studies must be performed on-site to really understand the design context (see Fig. 1) and to have close contact with the system architects and their managers responsible for solving the 'hot' issue. It also accelerates obtaining information to calibrate and validate performance models. We therefore advocate the 'industry-as-laboratory' approach [19].

When model-based engineering has shown (substantial) measurable successes by resolving performance issues effectively and efficiently, system designers may

either become eager to learn how it works or show resistance due to feeling threatened in no longer being the one expert capable of solving those issues (with traditional approaches). Although most academic work stops after successfully performing a case study, this is actually the moment where the cultural change in the way of working starts. Supported by their management, system architects should be able to adopt the new way of working in terms of applying the academic approaches themselves: the domain experts also need to become experts on performance modeling *their own* systems. Next to some minimal form of educating system architects on the new techniques and tools, introducing DSLs is a crucial aspect of this phase. The ability to automatically generate performance models from reverse-engineered DSLs as described in Sect.2, where design alternatives are mostly still explored manually defines maturity level 2. Maturity level 3 supersedes level 2 when proper embedding of performance prediction in the industrial design process is realized such that performance are indeed taken into account right from the start of the design and system architecting phases.

Maturity level 3 marks a predictive attitude towards performance instead of a reactive one. It may also be the highest achievable maturity level when state-of-the-art academic approaches do not (yet) provide solutions to further automate the design process. In some cases, especially when the intrinsic complexity[4] of the domain (most prominently determined by the amount of dynamism in systems) is relatively low, academic work does provide some approaches suitable for adoption by industry. An example is the static-order scheduling of loop-control functionality on (multi-core) processors in the lithography systems of ASML (see Sect. 4). Maturity level 4 is achieved when the exploration of alternative designs is supported with (semi-)automatic tooling. Maturity level 5 classifies the situation where the creative insights of system architects have been completely captured in a fully automated design process that realizes 'performance-by-construction'.

As mentioned above, some form of education is required in order to achieve adoption by industry. We experienced that it is important to distinguish the roles of a user of the new approach and of those who will maintain the corresponding design tooling with any relevant support. Such maintenance and support is crucial in an industrial setting. Where users shift to applying performance modeling and analysis, educating the developers of design tools in understanding the exploitation of the formal methods hidden behind the user-friendly DSLs and the DSL-based infrastructure is an even bigger challenge. We observe that different industries address this challenge differently, ranging from outsourcing to (small-scale) in-house education programs that focus on how to make performance models for the specific domain and to maintain the design tooling (DSLs with their generation algorithms) instead of 'coding implementations'.

[4] The sheer size of systems (in the number of parts) is a different kind of complexity.

4 Industrial Applications

We briefly report on three different application domains, where the approach described in Fig. 3 has been or is being introduced into industry.

4.1 Lithography Systems (ASML)

ASML develops lithography systems for the semiconductor industry. To achieve the required nanometer precision in combination with high throughput, these systems consist of hundreds of sensors and actuators which are controlled by thousands of control tasks. The development of such a system requires multi-disciplinary trade-offs involving mechatronic engineers, electronic engineers and software engineers. An early system-wide insight in timing bottlenecks is crucial to avoid costly redesigns and to meet time-to-market and quality constraints.

The loop-control systems are developed using a reference architecture that is captured by a coherent collection of inter-dependent DSLs [21]. These DSLs are used to specify the system according to the Y-chart modeling paradigm, where the origin of the application, platform and mapping lies in different departments.

- The application defines the mechatronic control logic, including networks of so-called servo groups and transducer groups.
- The platform level is captured by DSLs to describe electronic hardware and their physical architecture consisting of high performance multi-core processors, IO boards, and switched interconnection networks. Also physical limitations, such as the maximal frequency of an IO board, can be expressed.
- The mapping describes static deployment of application elements onto the platform resources. In addition, it also configures certain timing synchronization aspects of the platform.

The approach relies on a constructive scheduling algorithm to compute a static order schedule for each processor core such that the latency requirements of the loop-control systems are met [3]. To able to compute these schedules, a number of decisions have to be taken about the platform configuration, such as the number of processors, the topology of the switched interconnects and the setting of synchronization timers. To support these decisions, the DSLs are transformed into an executable POOSL model which is also structured according to the Y-chart paradigm. It formalizes the total system in terms of stochastically-timed communicating parallel processes. During simulation of the POOSL model, performance requirements of the applications are checked automatically. Moreover, the user obtains information about jitter, timing averages, and the load of processors and communication switches.

As reported in [30], the performance of the POOSL simulator easily deals with 4000 control tasks. The POOSL models were calibrated using an existing embedded control system. For a new system release, various significant performance improvements were identified. Important for the industrial adoption of the performance analysis techniques is the automatic generation of analysis models. Moreover, there is a strong coupling with synthesis, because the DSLs are also used for the generation of schedulers and code.

4.2 Multi-functional high-end printer (Océ)

In projects with the company Océ the performance of high-end printers has been addressed. An important performance measurement is the throughput as perceived by the user, i.e., the number of pages printed per minute. Part of the throughput is determined by the data processing part which has to deal with different types of jobs, such as scan jobs and copy jobs. Each job has a number of characteristics which determine the processing steps to be taken. For instance, the paper size (A3, A4, ...), the number of images per paper sheet, and the export format for scan jobs (pdf, jpeg, ...). Since this involves large size bit maps, there are also compression and decompression steps.

Designers of the data processing part have to make a large number of decisions about the processing units to use (CPUs, FPGAs, ..) and the type and size of memory. Many questions have to be answered, e.g., whether certain jobs are allowed to execute in parallel or whether certain steps may share a certain resource. Since costs are an important aspect, designers have to investigate whether the required performance can be achieved with reduced hardware and which performance can be achieved on a certain hardware platform.

To address these performance questions, a DSL for data processing architectures has been devised [24]. To achieve a separation of concerns, the DSL is structured in the three levels of the Y-chart paradigm.

- The application level contains a number of use cases, such as scanning a stack of pages and printing it three times double-sided. A use case consists of a number of atomic steps and a partial order on these steps.
- The platform level uses three types of resources: memory (with a certain capacity), bus (with a maximum bandwidth), and executor (with a processing speed). Connections between these resources limit the data flow.
- The mapping level specifies which steps run on which executors and which data is stored where.

Having the data processing architecture represented in a DSL allows an easy translation to various analysis tools with different capabilities. For instance Uppaal [5] has been used, because it allows exhaustive model checking of timed automata. However, for large models the time needed for the analysis became too large. Hence, a dedicated simulator, tuned to this particular domain, has been built to perform large scale simulations [12]. This allows numerous simulation runs over multiple design, for a fast identification of bottlenecks and promising designs.

4.3 Interventional X-ray Systems (Philips)

Performance aspects of interventional X-ray systems are addressed in a project with Philips. These systems are used for minimally invasive treatments. The number of medical procedures is quickly increasing and includes cardiovascular applications, e.g., placing a stent via a catheter, neurology and oncology. In these procedures, the surgeon is guided by X-ray images which have been processed

extensively to obtain a clear image of the essential medical structures using a minimal amount of X-ray.

For the eye-hand coordination of the surgeon, there are clear upper bounds on the latency and the jitter of the image processing chain. However, experiments with surgeons show that a small percentage of deadline misses is allowed.

For every new release, the designers have to make a careful trade off between the required X-ray dose, the hardware configuration (typically PFGAs and multi-core PCs), the algorithms to use, the order and the allocation of processing steps, the image quality, and the supported image resolutions. For instance, to reduce costs and maintenance there is a wish to reduce the number of PCs in the system, but it is very difficult to reason about the impact on the performance.

To support the designers of the image chain, the iDSL approach has been developed [27], consisting of a DSL for service systems, a mapping to the MODEST toolset [10, 11], and visualization tools. The iDSL approach supports a high level description of the image processing chain, with the three levels of the Y-chart paradigm. Moreover, three additional aspects can be specified:

- A scenario which restricts the performance analysis to a particular set of service requests. E.g., for the image processing, we can specify assumptions about the arrival rate of raw images from the detector.
- A measure defines the required performance observations and how to obtain them. E.g., a cumulative distribution function (CDF) obtained via model checking or average response times by means of simulation.
- A study allows the definition of a number of design instances that have to be evaluated.

The iDSL tools translate the DSL specification to a number of models in MODEST. Next the iDSL tools call the MODEST tools iteratively to obtain a particular performance observation. For instance, a number of simulation runs or repeated calls to a model checker to construct a CDF. There is a high degree of automation which makes it easy to switch between model checking and simulation.

Performance also plays a crucial role in the movement control part of an interventional X-ray system. This concerns, for instance, movements of patient table and the C-arm on which generator and detector are mounted. In our project we have addressed the redesign of the collision prevention component [17]. A DSL has been developed to express the rules of collision prevention at a high level of abstraction. Next a number of transformations have been defined to models for simulation, formal verification and performance analysis. The performance analysis of collision prevention concentrates on the computations needed to compute distances between objects [28]. It uses a generated POOSL model to perform simulations and to obtain statistics about expected execution times. The model uses performance profiles of the basic computation steps. Moreover, it has been calibrated using performance measurements on an existing legacy component. Finally, code has been generated from the same DSL, which was an important factor for the industrial adoption of the approach.

5 Related Work

An extensive overview of research that aims at the integration of performance analysis in the software development process can be found in [4]. This survey discusses methodologies based on queueing networks, process algebra, Petri nets, simulation methods, and stochastic processes. To achieve integration, many methods are connected to existing notations which describe software designs, often based on UML. More recently, [7] reported on the design of an ultra-modern satellite platform using the AADL notation in combination with several formal modeling and analysis techniques, including performance analysis. In our work, we do not start from these general purpose modeling languages, because usually such models are not available at the right level of abstraction. Instead, we use DSLs and transformations to suitable performance models.

Traditionally, DSLs have been developed for the construction and maintenance of software systems [29, 16]. The construction of DSLs has been facilitated by new technology such as the Eclipse Modeling Framework (EMF) [23] which is supported by a large collection of open source tools, including EMFText [22], Xtext [2], and Acceleo [1]. With such tools, it becomes relatively easy to generate different artefacts from a DSL. Relevant for our approach is the possibility to generate analysis models and code from the same DSL instance [6]. This turns out to be an important factor for industrial adoption. Note, however, that our focus is not on software only, but we consider a broader multi-disciplinary system scope.

Methodologies based on the Y-chart paradigm [14] can be found in [15] which compares a number of models of computation and three methodologies: Metropolis, Modular Performance Analysis (MPA), and Software/Hardware Engineering (SHE) supported by POOSL. An application of MPA with the Real-Time Calculus [26] to a distributed in-car radio navigation system has been described in [31]. Related tool support for this approach is provided by SymTA/S [9]. A comparison of these techniques with, e.g., analysis based on timed automata using Uppaal [5], can be found in [18].

6 Concluding remarks

Projects with industry on performance analysis revealed a number of needs:

- Support for system-wide trade-offs that involve multiple disciplines and takes business and market aspects into account.
- Tools that allow the fast generation of a large number of approximate results for large-scale systems. These results should correspond to industrially relevant questions.
- Convenient calibration of analysis results based on measurements of existing systems.
- Light-weight modeling where it is easy to change software design, hardware, and deployment.

– Guidelines about which analysis method to use for which purpose. It should be easy to switch between methods.

Our approach is based on DSLs to capture the essential domain knowledge in a concise and readable way. By defining appropriate transformations, formal analysis models can be generated automatically. Together with the use of the Y-chart paradigm this leads to a clear separation of concerns and a framework which allows changes easily and supports design space exploration. Adding a new analysis method is relatively easy by defining an additional generator. To get fast results for large-scale systems, we often use simulation tools that are based on formal methods. Visualization tools present the results in terms of the domain.

This approach is most effective when the performance models are generated from DSLs which are also used for the functional design. A close connection with the reference architecture and models from which code is generated stimulates industrial acceptance and the incorporation into the industrial work flow. Nevertheless, reaching a high level of maturity (see Sect. 3), such as achieved at ASML, requires many years of close collaboration.

References

1. Acceleo. http://www.eclipse.org/acceleo/, 2015.
2. Xtext. http://www.eclipse.org/Xtext/, 2015.
3. S. Adyanthaya, M. Geilen, T. Basten, R. Schiffelers, B. Theelen, and J. Voeten. Fast multiprocessor scheduling with fixed task binding of large scale industrial cyber physical systems. In *2013 Euromicro Conference on Digital System Design, DSD 2013, Los Alamitos, CA, USA, September 4-6, 2013*, pages 979–988, 2013.
4. S. Balsamo, A. di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: a survey. *Software Engineering, IEEE Transactions on*, 30(5):295–310, 2004.
5. G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *Formal Methods for the Design of Real-time Systems*, volume 3185 of *LNCS*, pages 200–236. Springer, 2004.
6. G. Edwards, Y. Brun, and N. Medvidovic. Automated analysis and code generation for domain-specific models. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 161–170, 2012.
7. M.-A. Esteve, J.-P. Katoen, V. Y. Nguyen, B. Postma, and Y. Yushtein. Formal correctness, safety, dependability, and performance analysis of a satellite. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 1022–1031. IEEE Press, 2012.
8. K. Forsberg and H. Mooz. The relationship of system engineering to the project cycle. *INCOSE International Symposium*, 1(1):57–65, 1991.
9. A. Hamann, R. Henia, R. Racu, M. Jersak, K. Richter, and R. Ernst. SymTA/S - Symbolic Timing Analysis for Systems. In *Work In Progress session - Euromicro Workshop on Real-time Systems*, 2004.
10. A. Hartmanns. MODEST - a unified language for quantitative models. In *The 2012 Forum on Specification and Design Languages (FDL)*, pages 44–51. IEEE, 2012.

11. A. Hartmanns and H. Hermanns. The MODEST toolset: an integrated environment for quantitative modelling and verification. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 8413 of *LNCS*, pages 593–598. Springer Berlin Heidelberg, 2014.
12. M. Hendriks, T. Basten, J. Verriet, M. Brassé, and L. Somers. A blueprint for system-level performance modeling of software-intensive embedded systems. *International Journal on Software Tools for Technology Transfer*, pages 1–20, 2014.
13. M. Hendriks, J. Verriet, T. Basten, B. Theelen, M. Brassé, and L. Somers. Analyzing execution traces – critical path analysis and distance analysis. *Submitted to: Software Tools for Technology Transfer*, 2015.
14. B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf. An approach for quantitative analysis of application-specific dataflow architectures. In *ASAP '97: Proc. of the IEEE International Conference on Application-Specific Systems, Architectures and Processors*, page 338. IEEE Computer Society, 1997.
15. J. Lapalme, B. Theelen, N. Stoimenov, J. Voeten, L. Thiele, and E. M. Aboulhamid. Y-chart based system design: a discussion on approaches. In *Nouvelles approches pour la conception d'outils CAO pour le domaine des systems embarqu'es*, pages 23–56. Universite de Montreal, 2009.
16. M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Computing Surveys*, 37(4):316–344, 2005.
17. A.J. Mooij, J. Hooman, and R. Albers. Early fault detection using design models for collision prevention in medical equipment. In *Foundations of Health Information Engineering and Systems (FHIES 2013)*, volume 8315 of *LNCS*, pages 170 – 187. Springer Berlin Heidelberg, 2014.
18. S. Perathoner, E. Wandeler, and L. Thiele. Evaluation and comparison of performance analysis methods for distributed embedded systems. Technical report, ETH Zurich, Switzerland, 2006.
19. C. Potts. Software-engineering research revisited. *IEEE Softw.*, 10(5):19–28, 1993.
20. A. Sangiovanni-Vincentelli and G. Martin. Platform-based design and software design methodology for embedded systems. *IEEE Des. Test*, 18(6):23–33, 2001.
21. R. Schiffelers, W. Alberts, and J. Voeten. Model-based specification, analysis and synthesis of servo controllers for lithoscanners. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, MPM '12, pages 55–60, New York, NY, USA, 2012. ACM.
22. Software Technology Group, TU Dresden. EMFText. http://www.emftext.org/, 2015.
23. D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *Eclipse Modeling Framework*. Pearson Education, 2008.
24. E. Teeselink, L. Somers, T. Basten, N. Trcka, and M. Hendriks. A visual language for modeling and analyzing printer data path architectures. *Proc. ITSLE*, 20, 2011.
25. B. Theelen, O. Florescu, M. Geilen, J. Huang, P. van der Putten, and J. Voeten. Software/hardware engineering with the parallel object-oriented specification language. In *Proceedings of the 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign*, MEMOCODE '07, pages 139–148, Washington, DC, USA, 2007. IEEE Computer Society.
26. L. Thiele, S. Chakraborty, and M. Naedele. Real-time calculus for scheduling hard real-time systems. In *Proc. IEEE International Symposium on Circuits and Systems*, volume 4, pages 101–104, 2000.
27. F. van den Berg, A. Remke, and B. Haverkort. A domain specific language for performance evaluation of medical imaging systems. In *Proceedings of the 5th Workshop on Medical Cyber-Physical Systems, MCPS 2014, Berlin, Germany*, volume 36

of *OpenAccess Series in Informatics (OASIcs)*, pages 80–93, Dagstuhl, Germany, 2014. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
28. F. van den Berg, A. Remke, A. Mooij, and B. Haverkort. Performance evaluation for collision prevention based on a domain specific language. In *Computer Performance Engineering*, volume 8168 of *LNCS*, pages 276–287. Springer Berlin Heidelberg, 2013.
29. A. van Deursen, P. Klint, and J. Visser. Domain-specific languages: an annotated bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.
30. J. Voeten, T. Hendriks, B. Theelen, J. Schuddemat, W.T. Suermondt, J. Gemei, K. Kotterink, and C. van Huët. Predicting timing performance of advanced mechatronics control systems. In *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pages 206–210, 2011.
31. E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse. System architecture evaluation using modular performance analysis: a case study. *International Journal of Software Tools for Technology Transfer (STTT)*, 8(6):649–667, 2006.