Model-driven system-performance engineering for cyber-physical systems

van der Sanden, B.; Li, Y.; van den Aker, J.; Akesson, B.; Bijlsma, T.; Hendriks, M.; Triantafyllidis, K.; Verriet, J.; Voeten, J.; Basten, T.

DOI
10.1145/3477244.3477985

Publication date
2021

Document Version
Final published version

Published in
2021 International Conference on Embedded Software : EMSOFT 2021

License
CC BY

Model-Driven System-Performance Engineering for Cyber-Physical Systems

Industry Session Paper

Bram van der Sanden[1], Yonghui Li[1], Joris van den Aker[1], Benny Akesson[1,3], Tjerk Bijlsma[4], Martijn Hendriks[1], Kostas Triantafyllidis[1], Jacques Verriet[1], Jeroen Voeten[2], Twan Basten[2,1]
[1]ESI (TNO), Eindhoven, Netherlands
[2]Eindhoven University of Technology, Eindhoven, Netherlands
[3]University of Amsterdam, Amsterdam, Netherlands
[4]DEMCON, Eindhoven, Netherlands

ABSTRACT

System-Performance Engineering (SysPE) encompasses modeling formalisms, methods, techniques, and industrial practices to design systems for performance, where performance is taken integrally into account during the whole system life cycle. Industrial SysPE state of practice is generally model-based. Due to the rapidly increasing complexity of systems, there is a need to develop and establish *model-driven* methods and techniques. To structure the field of SysPE, we identify (1) industrial challenges motivating the importance of SysPE, (2) scientific challenges that need to be addressed to establish *model-driven* SysPE, (3) important focus areas for SysPE and (4) best practices. We conducted a survey to collect feedback on our views. The responses were used to update and validate the identified challenges, focus areas, and best practices. The final result is presented in this paper. Interesting observations are that industry sees a need for better design-space exploration support, more than for additional performance modeling and analysis techniques. Also tools and integral methods for SysPE need attention. From the identified focus areas, scheduling and supervisory control is seen as lacking established best practices.

CCS CONCEPTS

• **General and reference → Performance**; *Design*; Empirical studies.

KEYWORDS

System-performance engineering, model-driven design, CPS

ACM Reference Format:
Bram van der Sanden, Yonghui Li, Joris van den Aker, Benny Akesson, Tjerk Bijlsma, Martijn Hendriks, Kostas Triantafyllidis, Jacques Verriet, Jeroen Voeten, Twan Basten. 2021. Model-Driven System-Performance Engineering for Cyber-Physical Systems: Industry Session Paper. In *2021 International Conference on Embedded Software Companion (EMSOFT'21 Companion), October 8–15, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3477244.3477985

EMSOFT'21 Companion, October 8–15, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8712-5/21/10.
https://doi.org/10.1145/3477244.3477985

1 INTRODUCTION

> **System performance** – *the amount of useful work done by a system - measured in production speed of products of a predefined quality.*
>
> **System** – *a group of interacting interdependent elements forming a unified whole.*

System performance often brings the competitive advantage for high-tech Cyber-Physical Systems (CPS) like semiconductor equipment, production printers, analytical instruments, and medical equipment. To meet market demands for product quality, product customization, and total cost of ownership per product, systems need to meet ever more ambitious performance targets relating to system productivity. Performance is a cross-cutting system-level concern, with intricate relations to other system-level concerns like product quality, cost, reliability, security, and customizability.

Designing for performance implies that system performance is a first-class citizen integrally taken into account during the full system life cycle. **System-Performance Engineering (SysPE)** encompasses modeling formalisms, methods, techniques, and industrial practices to design for performance. In this paper, we position SysPE as a field of study. We primarily target the domain of CPS, with a focus on single systems, i.e., CPS with a single managerial and operational scope of control (following Maier's differentiation between systems and systems of systems (SoS) [44]). The expressed views and insights may be useful in other domains than CPS and they extend to SoS. However, performance-engineering challenges, focus areas, and best practices in other domains and for SoS may differ to some extent in content and importance.

SysPE in today's industrial practice is typically *model-based*, with models being used to support design decisions. We advocate **Model-Driven System-Performance Engineering (MD-SysPE)** to systematically address SysPE challenges in industry. In model-driven development, models are pivotal. Models act as single source of truth and they form a basis for the automated synthesis of implementation artifacts. In MD-SysPE, for instance, schedules, schedulers, or controllers may be synthesized from models, guaranteeing performance by construction. MD-SysPE is essential to make the right design decisions during early stages of system development and to optimize performance during system operation. Early insight in system performance improves time-to-quality by requiring less rework in later stages of development. Designing for performance improves the cost-performance ratio of the final product by minimizing system over-dimensioning. Furthermore, it enables a wider range of system variants and operating conditions by taking into account system variability and context during design and operation.

To structure the field of SysPE, we identify (1) industrial challenges motivating the importance of MD-SysPE, (2) scientific challenges that need to be addressed to realize MD-SysPE; (3) focus areas for SysPE covering the entire system life cycle from development to operation, and (4) best practices for each of these focus areas. An initial overview was based on our experiences in almost 20 years of public-private partnership programs on performance engineering in the Dutch high-tech ecosystem, involving ESI (TNO) and both industrial and academic partners in the CPS domain. We then conducted a survey to collect feedback from the international industrial and academic community on the focus areas and best practices and to validate our views. We also gathered input on the state of practice and future challenges in SysPE. The received input and feedback has been integrated in the presented SysPE overview.

In the next section, we elaborate the context in which we position SysPE. We then consider the challenges in SysPE in Sec. 3, from both industrial and scientific perspectives. From these challenges, we identify five focus areas for SysPE in Sec. 4, that together cover the full system life cycle. For each of these areas, Sec. 5 then provides an overview of best practices. Sec. 6 highlights the importance of tool support and integral methods for MD-SysPE, identifying tooling and methods as an additional focus area, orthogonal to the life-cycle-related focus areas of Sec 4. Sec. 7 presents the survey methodology. In Sec. 8, we summarize the most important results from the survey. We moreover explain how the survey results affected the views expressed in the earlier sections. Sec. 9 concludes.

2 CONTEXT

The SysPE challenges, focus areas, and best practices presented in this paper specifically address the performance and system perspectives of CPS. However, systems operate in and interact with an environment, and performance cannot be considered in isolation. In this section, we briefly chart the context in which we elaborate our views on SysPE, providing both the scope for our work and pointers for further reading.

Systems Engineering. Systems Engineering [38] is an established approach to enable the realization of successful systems. Systems engineering covers all relevant aspects from the needs of customers to the business needs of the supplier. This includes system performance. The trend is towards *Model-Based* (Systems) Engineering (MB(S)E) [37]. MB(S)E is a methodology that focuses on using domain models as primary means of communication, in contrast to document-centric engineering. Whereas in MBE, models play an important role in the engineering process, in *Model-Driven* Engineering (MDE) [10], models *drive* the process during the entire system life cycle, from requirements engineering to development, implementation, verification and validation, deployment and operation. Theelen [68] suggests to structure MB/DE methodologies in terms of formalisms, techniques, methods, and tools, a proposal that was later adopted in the BoDERC framework [31]. Models are expressed in formalisms, like automata, logic, and differential equations, or languages like SysML [27] or a domain-specific language (DSL) [25, 75]. Techniques are used to retrieve information from models, such as performance metrics, to optimize designs, or to generate code. Methods specify how to apply formalisms, techniques, submethods, and tools to address the design questions at hand. Tools support the application of formalisms, techniques, and methods. Throughout the paper, we use these methodology concepts to systematically present and express our views.

Empirical evidence supporting the success of MB/DE for complex systems such as CPS in practice is still inconclusive. A recent study [32] shows that claimed benefits are not yet supported by scientific literature, and more research is needed to formally *measure* the benefits. Another study [43] confirms *perceived* benefits of MBE in the embedded-systems domain through a survey among practitioners, but it also identifies challenges. In the current paper, we advocate MD-SysPE, and we present feedback collected from the industrial and academic community on the identified best practices. The feedback confirms the envisioned benefits of MD-SysPE, but also shows that many challenges remain, e.g., with respect to tooling. The latter is in line with a key conclusion from [43].

Software Performance Engineering. Software is an important part of any CPS. The need for explicit performance engineering in the software domain was recognized early, dating back to the early years of computing [60]. The software performance engineering (SPE) methodology, for instance, has evolved over several decades and provides formalisms, techniques, methods, and tools for optimizing software performance [61]. SPE promises that software designs and implementations will meet the performance requirements with a shorter development cycle. Techniques that are used, range from predictive modeling to the creation of a clear business case for performance engineering. The methodology fits well with an MDE approach and it links with performance monitoring [22], where data from system operation is used to manage and optimize performance. SPE has started to move into the CPS domain [62]. SysPE typically involves more disciplines than only software, giving rise to new challenges. However, SPE is more mature than SysPE for CPS and SysPE can benefit from established SPE practices.

Embedded Performance. Embedded systems are systems that are an integral part of other systems. They often comprise a combination of hardware and software and are used to control and/or monitor the system they are part of. As such, embedded-systems performance is crucial for the performance of any CPS. Modern embedded computing systems are high-performance systems that must meet stringent requirements related to (real-time) performance, power/energy consumption, and cost [77]. SysPE for CPS needs to integrally consider performance of the embedded computing systems at the heart of the CPS. Many aspects play a role in analyzing and optimizing performance of embedded systems, ranging from timing analysis [45] to resource allocation [59]. Relevant techniques from the embedded domain need to be integrated in MD-SysPE methods for CPS being developed to date.

Systems of Systems. Systems are often part of a bigger whole, operating in an environment with other systems and human users. In smart industry, warehousing, or intelligent transportation, for instance, CPS are combined into SoS such as a complete manufacturing plant, an automated warehouse shuttle system, or a vehicle platoon connected to an intelligent traffic infrastructure. Maier [44] postulated several characteristics of such SoS, of which managerial and operational independence are the most important ones. Performance of SoS depends not only on the systems themselves, but also on their interaction. Performance engineering for SoS therefore has specific challenges, beyond the challenges for SysPE for individual CPS. Examples are emergent behavior among interacting systems and evolutionary change of the SoS, as e.g. observed by Falkner et al. [21]. To cope with these additional challenges in optimizing performance of SoS, they propose an MDE solution combining high-level behavioral and workload modeling in 'what-if' simulations with generated executable deployments on prototype platforms. Such an approach complies very well with the SysPE focus areas and best practices we identify in this paper for individual CPS.

3 CHALLENGES IN DESIGNING FOR PERFORMANCE

For industry, system-level performance, typically in combination with other system-level KPIs, is a crucial business driver. The performance-engineering challenges that industry is facing lead to interesting scientific challenges. Solutions to those scientific challenges are needed to realize MD-SysPE and as such may have a high impact on industrial practice. We therefore summarize industrial challenges (IC) and scientific challenges (SC) in MD-SysPE. The identified challenges are based on our extensive experience in public-private collaborations involving SysPE, complemented with input received through the survey (as explained in Sec. 8.4). The challenges serve as motivation for the focus areas and best practices presented in later sections, and as an inspiration for researchers and developers active in SysPE. Addressing the challenges is crucial to establish MD-SysPE in industrial practice.

3.1 Industrial Challenges

In today's industrial practice, **performance problems often only materialize late (IC1)** in the development process of CPSs, in the prototyping or integration phases, or during system operation. Typically, system performance targets are broken down into performance targets and budgets for the key components in the system. Component and system prototypes are then used to evaluate whether performance targets are met. **The need for prototypes often leads to a costly and time-consuming iterative development process (IC2)** of updating and re-evaluating performance targets and budgets. Performance problems materialize during system operation when systems are used in **configurations or operating conditions that were not foreseen (IC3)** or not properly evaluated. The way in which the user or other systems interact with the system can have a significant impact on the system load and performance. Design-time **modeling and analysis need to resemble real use cases as close as possible (IC4)**. Accurate data about the system usage is needed to achieve this.

To avoid costly rework and system updates, it is essential to **consider system performance during early development (IC5)** and to **trace performance targets and budgets throughout system development and operation (IC6)**. To this end, industry needs **industrially-usable, domain-specific languages, methods, and tools (IC7)** to specify, model and analyze system-level performance early in the development process. These languages, methods and tools need to be usable by developers with different backgrounds, and **they need to cover all performance-related aspects of relevant disciplines (IC8)**, including for instance software, mechanics, and electronics. Models, methods, and tools are often company-specific, resulting in **limited re-use of methods and tools across industries (IC9)**.

Typical available performance-engineering solutions focus on individual disciplines at low abstraction levels. Models are also often limited to individual system components, because **system-level models are too complex to develop and analyze (IC10)** with existing methods and tools. **Traceability of performance aspects across levels of abstraction, across system components, and across disciplines is lacking (IC11)**.

Platform-based development approaches are used to tailor the system towards specific markets, introducing variation points in the design for specific configurations, modules, and options. This increased diversity makes it difficult to **keep a good overview of system performance across a product line (IC12)** without evaluating all system variants individually, and to identify the region of operation with good performance.

High performance computing is needed to deal with the increasing amounts of data that need to be processed. **Techniques are needed to allocate computations to a heterogeneous computing infrastructure and to analyze the impact of data processing on system performance (IC13)**, considering computational resources, network limitations, and communication overhead.

3.2 Scientific Challenges

To address the industrial challenges and to realize MD-SysPE in industrial practice, we need to tackle several scientific challenges. We need **performance models at high abstraction levels that are sufficiently accurate (SC1)** to support decision making during system development and operation. These models not only need to capture relevant performance aspects, but also the **relations and trade-offs with other system-level concerns (SC2)** like product quality, cost, reliability, security, and energy usage. We need **techniques and methods to analyze these models, to optimize designs for performance, and to explore trade-offs (SC3)** with other system-level concerns. **System variability should be integrally supported (SC4)** in modeling formalisms, analysis and optimization techniques, and performance-engineering methods.

Timing predictability (SC5) of systems and system components is a prerequisite to come to accurate system performance predictions. The performance models, techniques and methods need to **link to models, techniques and methods from all relevant disciplines (SC6)** that impact performance. This is particularly challenging because the various disciplines use discrete, continuous, and stochastic models. It is not straightforward to meaningfully combine analysis results for these different types of models. **Model consistency (SC7)** is essential in this context.

Modelling, analysis, and optimization need to scale (SC8) to the highly complex systems we see in the modern high-tech systems industry. We also need **solutions that work during system**

operation (SC9), under tight timing and resource constraints, to cope with different system configurations and changing operating conditions. We need **traceability of performance and performance optimization across abstraction levels, across disciplines, throughout system development, and during system operation (SC10)**. Further, **learning and evolving performance models from data (SC11)** in line with system operation and evolution is an important challenge. The ultimate goal is to **constructively synthesize and automatically adapt systems to optimize performance (SC12)**, starting from models. That is, to truly achieve MD-SysPE.

4 FOCUS AREAS

Given the industrial challenges presented in the previous section, we advocate MD-SysPE throughout the whole system life cycle to address these challenges. Domain-specific conceptual modeling techniques are needed that capture all relevant aspects across disciplines in a particular domain, such as production systems or networked systems. Models should provide an explicit overview of how different system components relate and interact, and how components impact performance and other relevant system qualities. Models should be the single source of truth. The models should link to analysis, synthesis, scheduling, and control techniques, enabling automated reasoning about the performance of design alternatives, constructive design-space exploration, and on-line performance optimization. Data collected during system operation provides feedback on operation and design. We identify five main focus areas for SysPE. In each of these areas, further research and development is needed to realize MD-SysPE:

(1) Performance architecting (PA)
(2) Model-driven design-space exploration (DSE)
(3) Performance modeling and analysis (PMA)
(4) Scheduling and supervisory control (SSC)
(5) Data-driven analysis and design (DDAD) (including data collection and model learning)

The five focus areas cover all aspects of system development and system operation that are relevant for MD-SysPE. They are largely independent of the chosen development process, and fit with, for instance, agile development and classical V-model development [24]. Fig. 1 positions the focus areas in the V-model system development process. System development starts with the requirements and architecting phase. Progressing to the system design and system implementation phases means detailing more specific system elements. After implementation of the needed components, the system is integrated, verified and validated. While the system is in operation, incremental development iterations may be performed to update the system or to adapt it to specific operating conditions, as illustrated by the small V-development iterations at the right.

Designing for performance starts with **performance architecting** (see, e.g., [31, 58]), to determine the performance aspects that need to be taken into account at the start of the development process and during the system life cycle, to balance those with other system-level concerns, and to ensure that the system (reference) architecture fits with the performance requirements.

In the early design phases, **model-driven design-space exploration** [50] is performed to explore trade-offs and find optimal designs within a given system architecture. Exploration is done at a high abstraction level, following pre-defined patterns to systematically cope with system complexity [29]. **Performance modeling and analysis** techniques [19, 39] are used to express and analyze the performance of specific system configurations. Techniques are targeted to the type of systems and performance requirements at hand, ranging from analytical modeling and reasoning about performance bounds to simulation and stochastic reasoning about performance. For CPS, modeling and analysis often needs to combine multiple modeling formalisms and analysis techniques to cover all relevant disciplines (see e.g., [12]). Fig. 1 positions PMA at the center of the development process. In practice, PMA plays a role throughout the entire process at all abstraction levels. The importance of PMA will only increase when moving from MB-SysPE towards MD-SysPE.

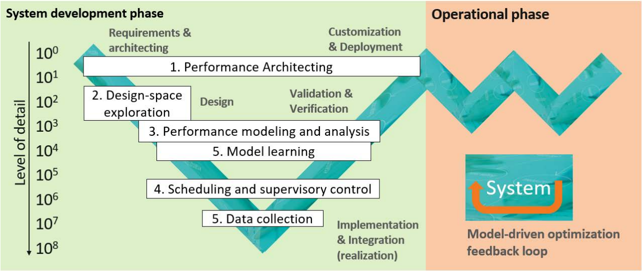

System development phase
Operational phase
Requirements & architecting
Customization & Deployment
Level of detail
10^0
10^1
10^2
10^3
10^4
10^5
10^6
10^7
10^8
1. Performance Architecting
2. Design-space exploration
Design
Validation & Verification
3. Performance modeling and analysis
5. Model learning
4. Scheduling and supervisory control
5. Data collection
Implementation & Integration (realization)
System
Model-driven optimization feedback loop

Figure 1: System development process with a positioning of the MD-SysPE focus areas

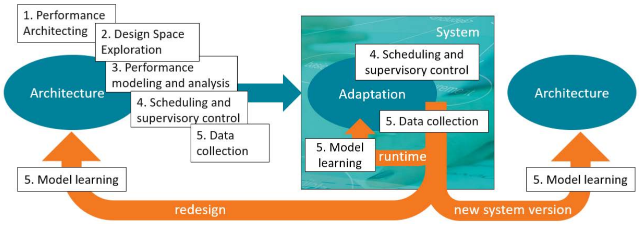

1. Performance Architecting
2. Design Space Exploration
3. Performance modeling and analysis
4. Scheduling and supervisory control
5. Data collection
Architecture
5. Model learning
System
4. Scheduling and supervisory control
Adaptation
5. Data collection
5. Model learning
runtime
Architecture
5. Model learning
redesign
new system version

Figure 2: System life cycle with a positioning of the MD-SysPE focus areas

Scheduling and supervisory control techniques [7, 52, 55] are essential to achieve the required performance during system operation. Schedulers and control strategies need to be designed during system development. Scheduling and control may then be optimized during system operation when the full operating conditions and all system inputs are known. Such on-line computations need to be done within strict time budgets and with the often limited processing resources available during system operation.

Accurate models are essential for all the mentioned activities. **Data-driven analysis and design** [9, 13, 65] techniques enable model learning [80], model validation [46], and model calibration [51]. Operational data can be used for monitoring performance targets during system operation [42], for diagnosing unexpected performance degradation [11, 48], for development of system updates, and for system (re-)design when developing new variants of the system at hand. Selecting the right data to be collected and lightweight, non-intrusive system instrumentation are essential [74].

Fig. 2 positions the focus areas in the system life cycle, emphasizing the feedback cycles from system operation to improve system performance. Operational data may be used to improve system performance at runtime and through system updates. It may also serve as valuable input for the development of new systems.

5 BEST PRACTICES

This section presents best practices for each of the five SysPE focus areas. The best practices reflect a combination of the industrial state of practice in SysPE (which is mostly model-based) and best practices that we envision as prerequisites to establish MD-SysPE. Sec. 8 discusses to what extent these best practices are recognized as established by industry and as important by academia. One of the best practices presented in this section (PA1) was added based on feedback from the survey participants; see Sec. 8.3 for details.

5.1 Performance Architecting

A *reference architecture* [15] is a prescribed template solution for the architecture of a concrete system at hand. Reference architectures are a commonly used best practice in systems engineering, in particular for product families and product lines. A reference architecture may prescribe, for instance, the structural decomposition of the system and the software architecture, as well as hardware and software interfaces. As first best practice for SysPE, we observe that it is important to **include a performance view in reference architectures (PA1)**. In manufacturing, for example, the Overall Equipment Effectiveness (OEE) method [1] takes into account availability, productivity, and quality of a manufacturing system. OEE computes the effectiveness of a system from the ratio between the realized value and expected value for each of these three aspects. OEE aligns closely with our notion of system performance defined as the production speed of products (productivity) of a predefined quality (quality).

Architecting commonly distinguishes *artifacts*, **domain models**, and **aspect models** [57]. Artifacts, such as documentation, code, and system data, describe the current system. Domain models generalize the essential domain concepts and their inter-relations beyond the scope of specific systems to the domain at hand, covering all relevant disciplines. The domain models link to aspect models that enable analysis of particular aspects of design alternatives. For performance architecting, **domain models (PA2) and aspect models (PA3) should make performance aspects explicit, including their relation to other system-level concerns** [6, 16, 78, 81].

Platform-based design (PA4) [56] and **budget-based design (PA5)** [26] form a basis for first-order system decomposition. Platform-based design targets the development of re-usable components, subsystems, and technology (comprehensively referred to as platforms). Budget-based design aims to budget critical aspects, including performance, and resources in a design. When integrated in a model-driven design flow, e.g., through virtual prototypes, platforms and budgeting help to speed up the development process, to better evaluate project risks, and to obtain better design trade-offs that explicitly consider performance during early design.

5.2 Model-Driven Design-Space Exploration

Model-based design-space exploration is the process of iterative **model-based prediction (DSE1) and validation (DSE2) of the performance of design alternatives** [39], to obtain feedback on the development process. Model-based DSE may help to reduce and refine the high-level design alternatives identified during architecting (e.g., through set-based design [58]) towards concrete designs. Model-based DSE is a step towards *model-driven* DSE (MD-DSE), in which models drive the development process and are the single source of truth. MD-DSE uses *virtual prototypes* for performance prediction and exploration of alternatives. In the development of system variants (product families, product lines), DSE may follow the **predict-the-past, explore-the-future paradigm (DSE3)** [36, 49]. The past is predicted by creating aspect models of existing systems or system components, calibrated or validated with measurements. This gives prediction accuracy and builds trust. To explore the future, performance aspects of design alternatives can be analyzed with the models adapted to these design alternatives.

To facilitate effective and efficient design-space exploration, it is important to **separate concerns** regarding system functionality and implementation aspects, and make the variation points explicit. The **Y-chart paradigm (DSE4)** [33, 40, 41] proposes to model application functionality and the implementation platform as separate elements, with an explicit mapping as variation point between them. This allows easy variation of application functionality, platform resources, and mapping choices and facilitates analyzing the performance impact of these choices. Design-space exploration systematically explores design alternatives around these variation points. Efficient DSE is difficult if alternatives require structural model adaptations that go beyond these explicit variation points.

5.3 Performance Modeling and Analysis

Performance modeling is typically done using a combination of knowledge-driven and data-driven modeling [20]. **Knowledge-driven modeling (PMA1)** builds on expert knowledge of domain specialists. **Data-driven modeling (PMA2)** creates models through regression or model learning from data collected from prototypes, tests, or system operation. Whereas domain models are typically developed during system architecting, performance aspect models are mostly developed and used during later phases, primarily design, but also for verification and validation. MD-SysPE requires models with rigorous mathematical foundations and tool support. Examples of such combinations are state charts [30] in StateFlow (mathworks.com/products/stateflow.html), timed stochastic decision processes [5, 67, 76] in POOSL [69] (poosl.esi.nl), and max-plus linear systems [18] in LSAT [72] (lsat.esi.nl).

A common technique to analyze performance of specific system behaviors is **simulation (PMA3)** [23, 47] (e.g., in Simulink, mathworks.com/products/simulink.html, or POOSL). Mourtzis [47], for instance, provides a nice survey on simulation for the design and operation of manufacturing systems, also covering the plant and manufacturing networks (i.e., the SoS) perspectives. **Analytical analysis (PMA4)** [18, 67] is used to derive performance estimates or bounds, e.g., in LSAT. **Model checking (PMA5)** [14], e.g., in UPPAAL [4] (uppaal.org), is used to exhaustively verify performance properties on a system model. Property verification can also be applied to execution or model traces (Gantt charts) built from actions, events, and signals, where timing properties are captured in formal logic as used for example in TRACE [34] (trace.esi.nl).

5.4 Scheduling and Supervisory Control

Supervisory controllers and schedulers need to realize correct system operation optimizing performance in relation to other system-level concerns like product quality and accuracy. Based on models of system behavior, (template code for) **schedulers and controllers can be synthesized (SSC1)** [3, 55]. Those schedulers and controllers should **optimize performance at runtime (SSC2)** [53] for varying system configurations and operating conditions. Runtime performance optimization through scheduling and control ranges from performance optimization for workloads being processed by the system [73] to model-driven quality- and resource management [35] to cope with configuration and operational changes in general, managing all relevant system qualities and resources. Schedulers and controllers should guarantee **performance by construction (SSC3)** [28]. For instance, a scheduler may guarantee a minimum productivity under varying operating conditions. Another example is a controller that minimizes energy usage at runtime taking into account operational information on energy usage while ensuring that the expected number of missed deadlines does not exceed 2%. SSC3 fits well with e.g. the Rigorous System Design (RSD) approach [8], which advocates model-driven *correctness by construction* in a broad sense. Performance is one of the mentioned aspects. Supervisory controller synthesis (SCS) [55] is a method to automatically synthesize a supervisor from a system model that restricts the system behavior to a given specification describing the allowed behaviors, hence guaranteeing correctness by construction. Performance is not explicitly optimized in SCS, but the resulting supervisor may be optimized for performance afterwards [64, 71]. A recent overview of scheduling theory and practice, covering both the scheduling of product flows and scheduling of software tasks on embedded processors, can be found in [7].

5.5 Data-Driven Analysis and Design

With the availability of large quantities of data and computing power, data-driven modeling, analysis, scheduling, and control complement their knowledge-based counterparts. Operational data may be used to improve system performance. It is essential to **collect the right data (DDAD1)**, via **light-weight, non-intrusive system instrumentation and timing measurements (DDAD2)** [63, 74] to minimize the impact on system performance. It is important to **ensure timing accuracy (DDAD3)** (e.g., via clock synchronization) and to consider **storage** and **bandwidth (DDAD4)** limitations that determine whether the required data can be collected in a real-time manner or not [79]. **Operational data can be related to models** through **model validation and calibration (DDAD5)** [51], **model learning (DDAD6)** [70, 80], or digital twinning [66], **balancing knowledge-driven and data-driven design approaches (DDAD7)** [54]. DDAD1 through 4 focus on the collection of data, whereas DDAD5, 6, and 7 focus on the use of data in modeling and design.

6 TOOLS AND METHODS

Any MDE methodology needs supporting tools and methods. The focus areas and best practices presented in the previous sections cover methods to some extent (albeit superficially), but do not cover the tooling perspective. Several responses to the survey that we conducted comment on tools and methods for SysPE, mostly on the lack of proper tooling and methods. Responses indicate the need for automation, e.g., in support of DSE, the lack of interoperability between tools, and the lack of integration in performance-engineering methods. This conforms to observations made in literature, e.g., [43]. In this section, we therefore briefly discuss the role and importance of tools and methods in MD-SysPE, concluding that **tool-supported MD-SysPE methods (TSM)** should be an additional cross-cutting focus area for MD-SysPE, complementing the earlier life-cycle-oriented focus areas identified in Sec. 4.

Tools are essential to support the efficient application of formalisms, techniques, and methods. Models can be expressed in DSLs [25, 75], tailored to the application domain, or in more generic languages like SysML [27] or Simulink [17]. By using tools, various engineering activities can be automated, including (domain-specific) model validation, analyzing performance of design alternatives, producing artefacts like documentation and code, evaluating trade-offs, and verification and validation. Design models can be (semi-)automatically refined into aspect models for a particular system aspect of interest, such as performance. Such aspect models can be analyzed, and the results can be taken into account in the design. This design-analysis feedback loop supports rapid DSE and helps in making the right design choices. To date, however, **mature, industrially usable tool-supported performance-engineering methods covering all relevant disciplines and the full life cycle of CPS** are lacking. Some languages and tools, like UML, SysML and Simulink, are in common use in industry in the CPS domain. Also, there is a plethora of academic formalisms, techniques, and tools that may be used in performance engineering. However, there are no systematic methods that connect these formalisms, techniques, and tools. The precise relations between models made with various tools and analysis results obtained from those models are often not clear, leading to potentially inconsistent results and conclusions. Traceability of performance aspects across abstraction levels, disciplines, and the system life cycle is lacking. These observations conform to the industrial and scientific challenges identified in Sec. 3. Therefore, we decided to introduce TSM as an additional cross-cutting focus area for MD-SysPE. It is orthogonal to the five focus areas of Sec. 4 in the sense that integral tool-supported methods are needed to support all these areas in a coherent manner, and across the full system life cycle. The development of model-driven methods and tool support for SysPE can only be done in close collaboration between industry and academia, where from industry both equipment manufacturers and tool providers need to be involved.

7 VALIDATION WITH THE ECOSYSTEM - SURVEY METHODOLOGY

As explained, the challenges, focus areas, and best practices presented so far are the result of many years of experience of the authors in public-private collaborations and a validation of the developed viewpoints with the SysPE community through a survey. This section describes the way in which we set up and executed the survey to collect feedback from the international industrial and academic performance-engineering community.

Survey Design. The survey was designed (in Survalyzer) as an anonymous cross-sectional study, where we collected responses in June 2021. The survey consisted of four parts:

(1) Company & personal profile;
(2) Performance engineering focus areas - your feedback;
(3) Best practices - your feedback;
(4) The future of performance engineering - challenges you see.

We used a combination of closed and open questions. Closed questions were used in Part 1 to collect characteristics about the respondent's organization, role and experience in SysPE. In Parts 2 and 3, we used closed questions to determine to which extent the focus areas and best practices are recognized as such and deemed important. We used open questions to allow respondents to give feedback on the focus areas and best practices. Part 4 used open questions to collect feedback on performance- and performance-engineering challenges and regarding the availability of methods/tools/techniques to address those challenges.

We tailored some questions to the respondent's role and organization to distinguish between respondents directly involved in SysPE, like architects and engineers, and those only indirectly involved, like university researchers.

- Role: System engineer/architect, engineer/designer, manager
  - To what extent do you recognize [focus area] in the daily practice of your company?
  - Do you think [focus area] gets the right level of attention in your organization?
- Role: Consultant, researcher, other
  - How important is [focus area] in your work?
  - Do you think [focus area] gets the right level of attention?
- Organization: System development organization, first or second tier industrial supplier, consulting firm, other
  - What system performance-related challenges do you foresee in the coming 5 years?
  - How are you planning to address these challenges?
- Organization: University, applied research organization
  - What new, or upcoming, scientific challenges do you see with respect to system-performance engineering of CPS?

Sampling Method. There is not yet an established network to reach the target audience of researchers and practitioners working on SysPE. We therefore used a combination of convenience sampling and snowball sampling. With convenience sampling, we reached out to the target population using the authors' combined networks, as well as to the academic and industrial parties that ESI is in regular contact with. We sent personalized invitations to this group, followed by one reminder two weeks later. We also sent the survey (and one reminder) to three mailing lists: EMSIG (announcements@lists.artist-embedded.org), HPC (hpc-announce@mcs.anl.gov), and Concurrency (concurrency@listserver.tue.nl). Finally, we distributed an invitation on LinkedIn. We applied snowball sampling by asking those invited to forward the survey to others who might be interested.

Data Selection. We received 87 fully completed responses, among several hundreds of started responses. Reasons to not complete the survey could be the survey length, or the fact that we used snowball sampling and social media, thereby targeting a broad audience. We observed that some participants filled in the questionnaire up to and including Part 3 on best practices, but did not continue with the open questions regarding the challenges in Part 4. We decided to include all responses where at least one of the questions regarding the best practices for a specific focus area were answered, yielding 100 responses considered in the survey analysis.

8 VALIDATION WITH THE ECOSYSTEM - SURVEY RESULTS

This section presents the results of the survey in four subsections in line with the survey sections. Table 1 summarizes the observations that can be made from the survey. All observations are based on a 95% confidence level, except when explicitly mentioned otherwise. When responses to specific questions are summarized, the number of participants X that responded is given as (n = X).

8.1 Profiles

From the profiling questions, we use the one on the type of organization that a respondent is employed with to differentiate between industry and academia.

QUESTION 1. *What type of organization do you work for? (n=100)*

System development org.	49%	1st or 2nd tier supplier	5%
University	35%	Applied research org.	4%
Consulting firm	6%	Other	1%

A respondent belongs to the **Industry** profile (60 respondents) if they indicate that their organization is a system development organization, 1st or 2nd tier supplier, or consulting firm, and to the **Academia** profile (35) if their organization is a university.

8.2 Focus Areas

We first present validation results for the focus areas. As explained in Sec. 7, we differentiated questions about focus areas for practitioners directly involved in SysPE and respondents only indirectly involved. Question 2 (see graphic at the end of this section) summarizes the aggregated answers to the survey questions checking whether the focus areas are recognized in daily practice, resp., considered important.

Four out of five focus areas are recognized by a (statistically significant) majority of the participants, meaning they answered either very recognizable/important or recognizable/important. PA is the most recognized area (81%), followed by PMA (71%), SSC (68%), DSE (62%), and DDAD (56%). The latter result is not statistically significant though. So we conclude that the respondents are undecided on DDAD. The observations on the validity of the focus areas are summarized in Table 1 as FA1.

In an open question, we asked whether any focus areas are considered missing. As already explained in Section 6, in part based on responses to the survey, we conclude that methods and tools should be a focus area for SysPE. The answers indicate that no other areas are missing. Several respondents mention concrete items in response to this question that fit within the identified best practices.

	Focus areas - overall
FA1	PA, DSE, PMA, and SSC are focus areas that are recognized/considered important[1] by the community; answers for DDAD are inconclusive. The orthogonal TSM focus area on tool-supported methods was added. No other areas are missing.
FA2	PA and SSC receive sufficient attention; DSE should receive more attention.
	Respondents are undecided about whether or not PMA and DDAD should receive more attention.
	Focus areas - industry vs. academia
FA.DSE	Industry respondents are inconclusive about the recognition of model-driven DSE in their daily practice, but they indicate that model-driven DSE should receive more attention.
	Academia indicates that DSE is important, but is inconclusive about whether or not it deserves more attention.
	Best practices - overall
PA	PA3, performance aspect models, and PA4, platform-based design, are recognized as best practices.
	Responses for PA2, domain models, PA5, budget-based design, are inconclusive.
	PA1, performance reference-architecture view, was added based on feedback (and hence not evaluated).
DSE	DSE1 and 2, model-based prediction, model validation, are recognized as best practices.
	DSE through the specific predict-the-past, explore-the-future (DSE3) and Y-chart (DSE4) methods is not recognized.
PMA	All best practices are recognized as such, except for model checking (PMA5).
SSC	SSC2, runtime optimization, is recognized as best practice; answers on the other two best practices are inconclusive.
DDAD	DDAD1 (data collection), 2 (instrumentation), 3 (timing accuracy), and 5 (calibration/validation) are recognized; responses for DDAD4 (storage/bandwidth), 6 (model learning) and 7 (combining data/knowledge) are inconclusive.
	Best practices - industry vs. academia
DSE34	The predict-the-past, explore-the-future paradigm is not recognized by academia (with inconclusive industry responses); the Y-chart is not recognized by industry (with inconclusive academic responses).
PMA1	Knowledge-driven modeling is recognized as best practice by industry, but not by academia (responses inconclusive).
SSC2	Runtime performance optimization is recognized by academia, whereas industry responses are inconclusive.
DDAD6	Model learning is not recognized as best practice by industry; it is significantly more often recognized by academia (where academia recognizes it as best practice with 90% confidence).

[1]The wording 'recognized' corresponds to a (statistically significant) aggregated score of at least 50% 'recognized'/'very recognized' answers in the corresponding survey question; 'important' corresponds to an aggregated score of at least 50% 'important'/'very important' answers; 'sufficient attention' corresponds to an aggregated score of at least 50% 'sufficient attention'/'too much attention' answers'; negative wordings correspond to cases where the two negative answer categories in the respective question exceed 50%.

Table 1: Most important observations from the survey results.

Quite a number of responses indicate that other aspects of systems (cost, correctness, quality, extensibility, dependability, security, and privacy) are also relevant. This is in line with, for instance, best practices PA2 and PA3 that state that performance models should cover the relation to other system-level concerns.

We also asked whether the focus areas receive sufficient attention in the ecosystem, see Question 3 and observation FA2 in Table 1. Two focus areas are considered to receive sufficient attention, namely PA and SSC. Interestingly, DSE is considered to receive insufficient attention. The respondents are undecided on PMA and DDAD. DSE is considered to be in need for extra attention by 64% of the respondents, followed by 47% for PMA, 42% for DDAD, 34% for PA, and 25% for SSC.

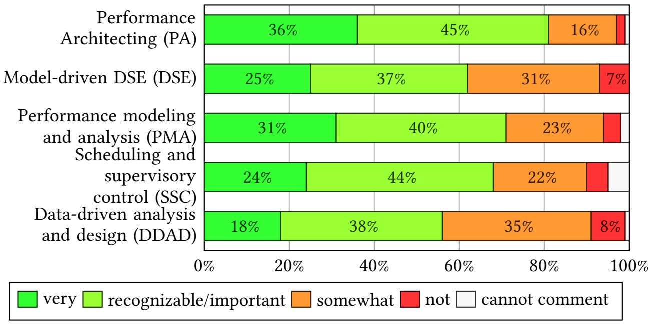
The community is very divided about DDAD, both in terms of recognition of the area and the need for extra attention. IC13, emphasizing the importance of data, was added as an industrial challenge based on the responses to the open questions in the survey (see Sec. 8.4), further illustrating the division. In line with IC13 (and SC11 on model learning and evolution), we believe that DDAD will only gain in importance with the increasing availability of data, not only in general for SE, but also specifically for MD-SysPE.

Given the observations summarized under FA1 and FA2, we investigated whether the industrial and academic communities differ in their views. When comparing academic and industrial responses, we observe that 52% of the industry respondents does not recognize MD-DSE in their daily work, whereas 48% does[1]. At the same time, 68% of the industry respondents indicates that it receives insufficient attention. A large majority of academic respondents indicates MD-DSE is an important area (83%), whereas they are inconclusive on whether or not it should receive more attention. FA.DSE in Table 1 summarizes these observations. The overall observation that MD-DSE deserves more attention is mainly driven by industry.

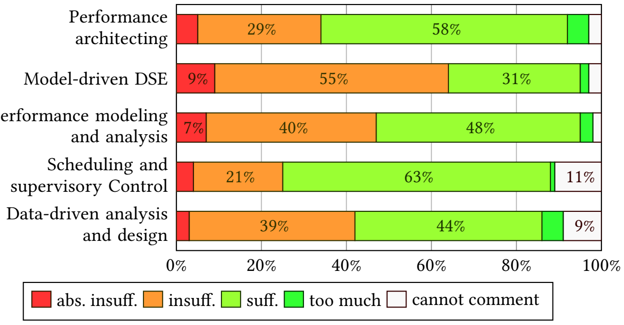
Another observation from the profiled data is that PMA is considered important by academia (85%) and receives insufficient attention (53%). Industry indicates that it is recognized in daily practice (64%) and receives sufficient attention (54%). The differences between academia and industry for PMA are not statistically significant though. So we may conclude that the two profiles do not significantly differ in their views on PMA. Also for the other three focus areas, PA, SSC, and DDAD, the results for industry and academia do not deviate (statistically significantly) from the overall observations for those areas.

[1]For space reasons, we do not include separate figures for the profiles, but summarize the most important findings in the text and in Table 1.

Question 2. *To what extent do you recognize [focus area] in the daily practice of your company? / How important is [focus area] in your work?* (n=100)



Question 3. *Do you think [focus area] gets the right level of attention (in your organization)?* (n=100)

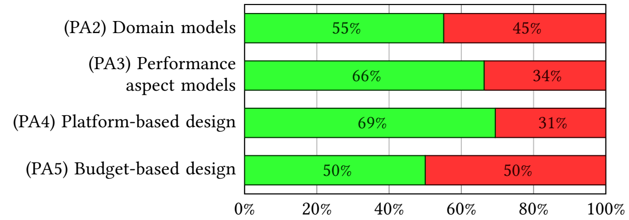


8.3 Best Practices

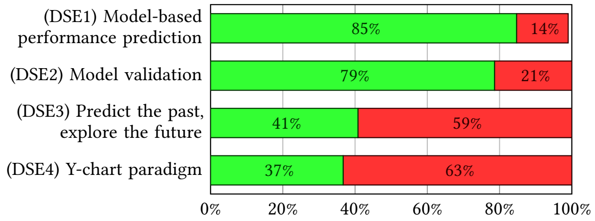
In this subsection, we discuss the survey results for the best practices presented in Sec. 5. For each focus area, we asked the participants whether or not they recognize the given best practices as such. We also asked in open questions per focus area whether any of the best practices is not considered industrially relevant. We wanted to explicitly check whether any potentially non-recognized best practices might be considered industrially not relevant. The latter is a stronger statement than not recognizing a best practice. Finally, we asked whether the participant sees any best practice not mentioned. As a result of feedback we received through these open questions, we added best practice PA1, stating that a reference architecture should have a performance view. Respondents commented, for instance, that performance should be taken into account in the system decomposition and in the software architecture. We did not add any further best practices. As with the open question regarding potentially missing focus areas, we also received several comments about missing best practices that we consider part of the already identified best practices. None of the identified best practices was considered not industrially relevant by any significant number of participants.

Performance Architecting. The results for PA are summarized in Question 4. PA1, which was added based on feedback, was not evaluated. From the other four best practices, PA3 and PA4 are recognized, whereas the results for the other two best practices are inconclusive. Entry PA in Table 1 summarizes these observations.

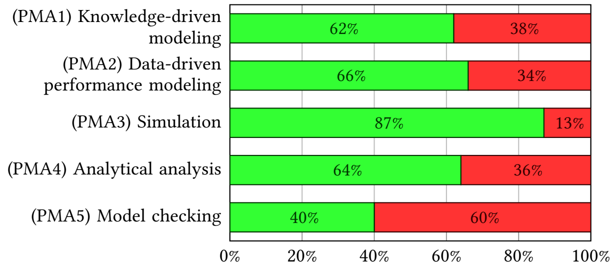
Question 4. *Which best practices regarding performance architecting do you recognize as such?* (n=98)



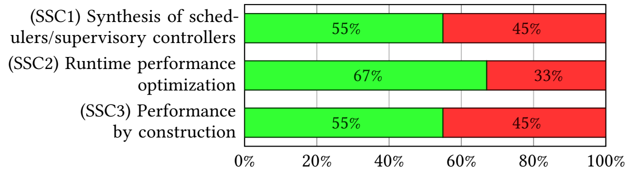
Question 5. *Which best practices regarding design-space exploration do you recognize as such?* (n=98)



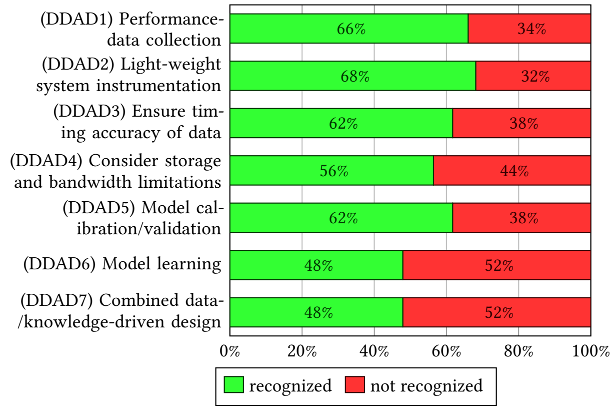
Question 6. *Which best practices regarding performance modeling and analysis do you recognize as such?* (n=100)



Question 7. *Which best practices regarding scheduling and supervisory control do you recognize as such?* (n=91)



Question 8. *Which best practices regarding data-driven analysis do you recognize as such?* (n=94)



We further analyze these results by looking at potential differences between industry and academia. Overall results for PA2 are inconclusive. It was hypothesized by one respondent that domain models (PA2) are mostly an academic practice, not appreciated by industry. It was also mentioned that domain models are at the same time relevant and not relevant and that creating domain models is difficult. 58% of industry respondents recognizes PA2 as a best practice, whereas only 46% of academia recognizes it. These differences are not statistically significant (with 95% confidence), but they certainly do not indicate that domain modeling is not recognized by industry. Also the overall response for PA5, budget-based design, is inconclusive. As for PA2, PA5 is recognized more by industry (58%) than by academia (39%). This difference is only significant with 90% confidence though. Finally, for PA3 and PA4 (which are both overall recognized), we see in both cases that the percentages of 'recognized' answers are similar between industry and academia (e.g., 67% for industry vs. 64% for academia recognizing PA3), but that these percentages are statistically significant for industry with 60 respondents and not significant for academia with 33 respondents[2]. This is a consequence of the relatively small number of academic respondents. Since the results are in line with the overall conclusions, we do not include them in Table 1 of the most important observations from the survey data. We moreover do not elaborate similar cases in the remainder.

Model-Driven Design-Space Exploration. Next, we summarize the results for the MD-DSE best practices; see Question 5. Model-based performance prediction (DSE1) and model validation (DSE2) cover a wide variety of techniques and are recognized as best practices. DSE3 and DSE4 correspond to specific methods using models for DSE. DSE4 is not recognized as a best practice. DSE3 is not recognized with 90% confidence (responses being inconclusive at 95% confidence).

Participants comment both positively and negatively on the Y-chart, mentioning for instance that it is not always clear how to separate the application from the platform. Automation of DSE, exploration of the predictable region of operation, and model transformations are mentioned as relevant aspects that need attention.

When looking at industry and academia, we observe that the predict-the-past, explore-the-future paradigm (DSE3) is not recognized by academia (71%) with industry being inconclusive (56% not recognizing it). The Y-chart (DSE4) shows the opposite pattern, with industry not recognizing it as best practice (73%) and responses from academia being inconclusive (53% not recognizing it).

Performance Modeling and Analysis. All PMA best practices are recognized as such, except for PMA5, model checking, which is not recognized; see Question 6. Simulation is most recognized as performance-analysis technique. Participants mention the flexibility and broad application domain as clear advantages. Regarding analytical analysis, participants comment that it is less widely applicable than simulation. For model checking, multiple participants mention limitations regarding scalability to industrial systems. Performance failure-mode analysis and sensitivity analysis are mentioned as important types of analysis. PMA should also consider the specific system use case scenarios.

An interesting observation is that PMA1, knowledge-driven modeling, is recognized as best practice by industry (77%), whereas responses from academia are inconclusive (37% recognizing PMA1 as a best practice). This difference between industry and academia is significant. We did not ask for clarification why a practice is or is not recognized, so the reasons for this difference would need further investigation. The result may indicate that techniques being developed by academia are not (yet) applied in practice, whereas industry has the domain knowledge to apply already established techniques.

Scheduling and Supervisory Control. For SSC, see Question 7, SSC2, runtime performance optimization, is recognized as best practice. Responses for the other two best practices are inconclusive. This is in line with the current state of practice that models are not yet driving the development process, as is required for SSC1 and SSC3.

Participants mention in their responses that using standard OS scheduling is an established practice. This is indeed true for software tasks, but it does not extend to the full CPS scope. Moreover, also OS schedulers may be configured through models and synthesis. Some participants comment that performance by construction, SSC3, is not realistic in practice. For complex CPS, realizing performance by construction is indeed challenging. However, for certain aspects or subsystems, it may be more easily realizable. A recent survey [2] shows, for instance, that establishing performance by construction is the second most common approach to ensuring that deadlines are met in real-time systems, after testing and checking for overruns.

When comparing industry and academia, we observe that optimizing performance at runtime is recognized by academia (79%) whereas industry responses are inconclusive (59% recognizing SSC2 as a best practice). In combination with the observation that also the other two best practices are inconclusive (also when considering industry and academia in isolation), we may conclude that industry lacks established best practices in this area.

Data-Driven Analysis and Design. For DDAD, see Question 8, DDAD1, 2, 3, and 5, all related to data collection, are recognized as best practices. Responses for DDAD4, taking storage and bandwidth into account, are inconclusive. Industry recognizes DDAD4 as best practice with 90% confidence and 61% positive responses. Academia is only 47% positive (and hence inconclusive). The use of data in modeling and design is a relatively recent development, captured by DDAD6, model calibration/learning, and DDAD7, hybrid data-/knowledge-driven design (including e.g. digital twinning). Model learning is not recognized as a best practice by industry, with 37% positive responses. Academia recognizes it as a best practice with 90% confidence and 66% positive responses. Two respondents comment that machine learning is, in their opinion, overrated.

Overall, we may conclude that *model-based* best practices are generally recognized as such, whereas practices closer towards *model-driven* SysPE are not always recognized (yet). Scheduling and supervisory control is lacking clearly established best practices. Academia recognizes more recent developments more often as best

[2] Although the data, see Sec. 8.1, has 35 respondents from academia, only 33 responded to this questions. All presented results for industry and academia are based on the respondents from those profiles that actually answered the question.

practice than industry, but it sometimes does not recognize well-established industrial best practices. We may conclude though that the identified best practices represent the state of practice in MB- and MD-SysPE quite well.

8.4 Challenges

In this final subsection, we summarize the responses to the open questions about the future of performance engineering and the challenges seen by the participants. In our analysis, we also included challenges mentioned in the answers to the open questions regarding focus areas and best practices. Based on these answers, we added three additional industrial challenges, namely IC4, IC12, and IC13, to the overview as presented in Sec. 3.

Several participants mention the challenge to resemble real use cases as close as possible in modeling and analysis. We captured this in challenge IC4. Considering specific answers given in this context, it was mentioned that the impact of the human user interacting with the system should be taken into account. It was also indicated that the availability of customer data is still limited, and that in-house modeling of use cases does not sufficiently resemble real usage. Furthermore, in the open questions related to focus areas and best practices, application-specific performance was mentioned, and that a domain model is not easy to define and verify since it depends on the application domain and the environment it works in. Techniques and methods from the DDAD focus area, in particular model learning and hybrid data-/knowledge-driven modeling may contribute to addressing these challenges.

Various participants also indicated the need to keep a good overview of system performance across a product line, captured as IC12. Participants mentioned that this becomes increasingly challenging with the increase in the number of system configurations and options. The challenge relates on the one hand to modeling, to make variation points explicit. On the other hand, it relates to analysis, involving both generic trade-off analysis, to determine when to stick to an established platform and when to deviate, and performance analysis over multiple configurations.

Finally, the survey participants mention that techniques are needed to allocate high-performance computations to a heterogeneous computing infrastructure and to analyze the impact of data processing on system performance, covered by IC13. High-performance computational tasks (e.g., image processing, digital twinning) are more and more often an integral part of CPS. Various challenges related to distributed heterogeneous systems were mentioned, including how to partition the application, mapping applications onto compute nodes, and how to analyze and ensure performance. Participants also indicated the increased importance of data, affecting the computational load, what platform is required, and the overall system performance.

9 CONCLUSIONS

This paper positions the field of SysPE by presenting industrial and scientific challenges, focus areas, and best practices. The presented views have been validated with the community through a survey. Industrial state of practice in SysPE is mostly model-based. A transition to model-driven SysPE is needed to cope with the ever increasing complexity of today's CPS. This transition requires a joint effort from industry and academia. The presented challenges and focus areas provide directions for further development.

The survey data is available via https://zenodo.org/record/5146160. The survey is still open for anyone interested: https://survey.tno.nl/nmlqoidkyf. Additional data (if sufficient and meaningful) will be made available through the above URL. It may also be used for follow-up publications.

REFERENCES

[1] I. P. S. Ahuja and J. S. Khamba. 2008. Total Productive Maintenance: Literature Review and Directions. *International Journal of Quality & Reliability Management* 25, 7 (2008), 709–756.

[2] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis. 2020. An Empirical Survey-based Study into Industry Practice in Real-time Systems. In *Proc. 2020 IEEE Real-Time Systems Symposium, RTSS'20*. IEEE, 3–11.

[3] K. Altisen, G. Goßler, A. Pnueli, J. Sifakis, S. Tripakis, and S. Yovine. 1999. A Framework for Scheduler Synthesis. In *Proc. 20th IEEE Real-Time Systems Symposium, RTSS'99*. IEEE, 154–163.

[4] G. Behrmann, A. David, K. G. Larsen, H. Håkansson, P. Pettersson, W. Yi, and M. Hendriks. 2006. Uppaal 4.0. In *Proc. 3rd Int. Conf. on the Quantitative Evaluation of Systems, QEST'06*. IEEE CS Press, 125–126.

[5] R. Bellman. 1957. A Markovian Decision Process. *Journal of Mathematics and Mechanics* 6, 5 (1957), 679–684.

[6] T. Bijlsma, B. van der Sanden, Y. Li, R. Janssen, and R. Tinsel. 2019. Decision Support Methodology for Evolutionary Embedded System Design. In *Proc. 2019 Int. Symposium on Systems Engineering, ISSE'19*. IEEE.

[7] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, M. Sterna, and J. Weglarz. 2019. *Handbook on Scheduling: From Theory to Practice* (second ed.). Springer.

[8] S. Bliudze, P. Katsaros, S. Bensalem, and M. Wirsing. 2021. On Methods and Tools for Rigorous System Design. *Software Tools for Technology Transfer* (2021), 1–6.

[9] J. Bosch. 2016. *Speed, Data, and Ecosystems – Excelling in a Software-Driven World.*

[10] M. Brambilla, J. Cabot, and M. Wimmer. 2017. *Model-Driven Software Engineering in Practice* (second ed.). Morgan & Claypool.

[11] A. Bunte, B. Stein, and O. Niggemann. 2019. Model-Based Diagnosis for Cyber-Physical Production Systems Based on Machine Learning and Residual-Based Diagnosis Models. *Proc. of the AAAI Conf. on Artificial Intelligence* 33, 01 (2019), 2727–2735.

[12] P. Carreira, V. Amaral, and H. Vangheluwe (Eds.). 2020. *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems.* Springer.

[13] S. R. Chhetri and M. A. A. Faruque. 2020. *Data-Driven Modeling of Cyber-Physical Systems using Side-Channel Analysis* (first ed.). Springer.

[14] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H.t Veith. 2018. *Model Checking.* MIT press.

[15] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone. 2010. The Concept of Reference Architectures. *Systems Engineering* 13, 1 (2010), 14–27.

[16] M. M. Cowing, M. E. Paté-Cornell, and P. W. Glynn. 2004. Dynamic Modeling of the Tradeoff Between Productivity and Safety in Critical Engineering Systems. *Reliability Engineering & System Safety* 86, 3 (2004), 269–284.

[17] J. B Dabney and T. L Harman. 2004. *Mastering Simulink.* Vol. 230. Pearson/Prentice Hall, Upper Saddle River.

[18] B. de Schutter and T. van den Boom. 2008. Max-Plus Algebra and Max-Plus Linear Discrete-Event Systems: An Introduction. In *Proc. 9th Int. Workshop on Discrete Event Systems, WODES'08*. IEEE CS Press, 36–42.

[19] P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli. 2012. Modeling Cyber–Physical Systems. *Proc. of the IEEE* 100, 1 (2012), 13–28.

[20] D. Dubois, P. Hájek, and H. Prade. 2000. Knowledge-Driven versus Data-Driven Logics. *Journal of Logic, Language and Information* 9, 1 (2000), 65–89.

[21] K. Falkner, C. Szabo, V. Chiprianov, G. Puddy, M. Rieckmann, D. Fraser, and C. Aston. 2018. Model-Driven Performance Prediction of Systems of Systems. *Software & Systems Modeling* 17 (2018), 415–441.

[22] S. Fischmeister. 2020. Mining Traces of Embedded Software Systems for Insights. In *Proc. of the ACM/SPEC Int. Conf. on Performance Engineering, ICPE'20*. ACM.

[23] G. S. Fishman. 2013. *Discrete-Event Simulation: Modeling, Programming, and Analysis.* Springer.

[24] K. Forsberg and H. Mooz. 1992. The Relationship of System Engineering to the Project Cycle. *Engineering Management Journal* 4, 3 (1992), 36–43.

[25] M. Fowler. 2010. *Domain-Specific Languages.* Addison-Wesley Professional.

[26] H. J. M. Freriks, W. P. M. H. Heemels, G. Muller, and J. H. Sandee. 2006. On the Systematic Use of Budget-Based Design. In *Proc. 16th INCOSE Int. Symposium.* Wiley.

[27] S. Friedenthal, A. Moore, and R. Steiner. 2014. *A Practical Guide to SysML: the Systems Modeling Language.* Morgan Kaufmann.

[28] M. R. Garey, R. L. Graham, and D. S. Johnson. 1978. Performance Guarantees for Scheduling Algorithms. *Operations Research* 26, 1 (1978), 3–21.

[29] M. Gries. 2004. Methods for Evaluating and Covering the Design Space During Early Design Development. *Integration* 38, 2 (2004), 131–183.

[30] D. Harel. 1987. Statecharts: A Visual Formalism For Complex Systems. *Science of Computer Programming* 8, 3 (1987), 231–274.

[31] M. Heemels and G. Muller (Eds.). 2007. *Boderc: Model-Based Design of High-Tech Systems.* Embedded Systems Institute.

[32] K. Henderson and A. Salado. 2021. Value and Benefits of Model-Based Systems Engineering (MBSE): Evidence from the Literature. *Systems Engineering* 24, 1 (2021), 51–66.

[33] M. Hendriks, T. Basten, J. Verriet, M. Brassé, and L. Somers. 2016. A Blueprint for System-Level Performance Modeling of Software-Intensive Embedded Systems. *Software Tools for Technology Transfer* 18, 1 (2016), 21–40.

[34] M. Hendriks, M. Geilen, A. R. B. Behrouzian, T. Basten, H. Ara Alizadeh, and D. Goswami. 2016. Checking Metric Temporal Logic with TRACE. In *Proc. 16th Int. Conf. on Application of Concurrency to System Design, ACSD'16*. IEEE, 19–24.

[35] M. Hendriks, M. Geilen, K. Goossens, R. de Jong, and T. Basten. 2021. Interface Modeling for Quality and Resource Management. *Logical Methods in Computer Science* 17, 2 (2021), 19:1–19:34.

[36] M. Hendriks, J. Verriet, T. Basten, M. Brassé, R. Dankers, R. Laan, A. Lint, H. Moneva, L. Somers, and M. Willekens. 2015. Performance Engineering for Industrial Embedded Data-Processing Systems. In *Proc. 16h Int. Conf. Product-Focused Software Process Improvement, PROFES'15*. Springer, 399–414.

[37] INCOSE. 2007. *Systems Engineering Vision 2020.* INCOSE Foundation.

[38] INCOSE. 2014. *A World in Motion - Systems Engineering Vision 2025.* INCOSE Foundation.

[39] R. Jain. 1991. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley.

[40] B. Kienhuis, E. Deprettere, K. Vissers, and P. van der Wolf. 1997. An Approach for Quantitative Analysis of Application-Specific Dataflow Architectures. In *Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures and Processors, ASAP'97*. IEEE, 338–349.

[41] J. Lapalme, B. D. Theelen, N. Stoimenov, J. Voeten, L. Thiele, and E. M. Aboulhamid. 2009. Y-chart Based System Design: a Discussion on Approaches. In *Nouvelles approches pour la conception d'outils CAO pour le domaine des systems embarqués*. Université de Montreal, 23–56.

[42] M. Leucker and C. Schallhart. 2009. A Brief Account of Runtime Verification. *The Journal of Logic and Algebraic Programming* 78, 5 (2009), 293–303.

[43] G. Liebel, N. Marko, M. Tichy, A. Leitner, and J. Hansson. 2018. Model-Based Engineering in the Embedded Systems Domain: an Industrial Survey on the State-of-Practice. *Software & Systems Modeling* 17 (2018), 91–113.

[44] M. W. Maier. 1998. Architecting Principles for Systems-of-Systems. *Systems Engineering* 1, 4 (1998), 267–284.

[45] C. Maiza, H. Rihani, J. M. Rivas, J. Goossens, S. Altmeyer, and R. I. Davis. 2019. A Survey of Timing Verification Techniques for Multi-Core Real-Time Systems. *Comput. Surveys* 52, 3 (2019), 1–38.

[46] S. Mitsch and A. Platzer. 2016. ModelPlex: Verified Runtime Validation of Verified Cyber-Physical System Models. *Formal Methods in System Design* 49 (2016), 33–74.

[47] D. Mourtzis. 2020. Simulation in the Design and Operation of Manufacturing Systems: State of the Art and new Trends. *Int. Journal of Production Research* 58, 7 (2020), 1927–1949.

[48] U. Odyurt, H. Meyer, A. D. Pimentel, E. Paradas, and I. Gonzalez Alonso. 2019. Software Passports for Automated Performance Anomaly Detection of Cyber-Physical Systems. In *Proc. Embedded Computer Systems: Architectures, Modeling, and Simulation. SAMOS'19*. Springer, 255–268.

[49] V. V. Parappurath, J. P. M. Voeten, and K. C. Kotterink. 2013. Calibration Error Bound Estimation in Performance Modeling. In *2013 Euromicro Conf. on Digital System Design, DSD'13*. IEEE, 97–102.

[50] A. D. Pimentel. 2017. Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration. *IEEE Design & Test* 34, 1 (2017), 77–90.

[51] A. D. Pimentel, M. Thompson, S. Polstra, and C. Erbas. 2008. Calibration of Abstract Performance Models for System-Level Design Space Exploration. *Journal of Signal Processing Systems* 50, 2 (2008), 71–77.

[52] M. L. Pinedo. 2016. *Scheduling - Theory, Algorithms, and Systems* (fifth ed.). Springer.

[53] K. Pruhs, J. Sgall, and E. Torng. 2004. Online Scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Chapter 15.

[54] R. Rai and C. K. Sahu. 2020. Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus. *IEEE Access* 8 (2020), 71050–71073.

[55] P. J. Ramadge and W. M. Wonham. 1987. Supervisory Control of a Class of Discrete Event Processes. *Journal on Control and Optimization* 25, 1 (1987), 206–230.

[56] A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi. 2004. Benefits and Challenges for Platform-Based Design. In *Proc. 41st Annual Design Aut. Conf., DAC'04*. ACM, 409–414.

[57] R. Schiffelers, Y. Luo, J. Mengerink, and M. van den Brand. 2018. Towards Automated Analysis of Model-Driven Artifacts in Industry. In *Proc. of the 6th Int. Conf. on Model-Driven Eng. and Software Dev., MODELSWARD'18*. SciTePress, 743–751.

[58] N. Shallcross, G. S. Parnell, E. Pohl, and E. Specking. 2020. Set-Based Design: The State-of-Practice and Research Opportunities. *Systems Engineering* 23, 5 (2020), 557–578.

[59] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak. 2017. A Survey and Comparative Study of Hard and Soft Real-Time Dynamic Resource Allocation Strategies for Multi-/Many-Core Systems. *Comput. Surveys* 50, 2, Article 40 (2017).

[60] C. U. Smith. 1986. The Evolution of Software Performance Engineering: A Survey. In *Proc. of 1986 ACM Fall Joint Computer Conf., ACM'86*. ACM, 778–783.

[61] C. U. Smith. 1990. *Performance Engineering of Software Systems.* Addison-Wesley.

[62] C. U. Smith. 2020. Software Performance Antipatterns in Cyber-Physical Systems. In *Proc. of the ACM/SPEC Int. Conf. on Performance Engineering, ICPE'20*. ACM, 173–180.

[63] D. B. Stewart. 2002. Measuring Execution Time and Real-Time Performance. In *Proc. of the Embedded Systems Conf., ESC SF'02.*

[64] R. Su, J. H. van Schuppen, and J. E. Rooda. 2011. The Synthesis of Time Optimal Supervisors by using Heaps-of-Pieces. *IEEE Trans. on Automatic Control* 57, 1 (2011), 105–118.

[65] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. 2018. Digital Twin-Driven Product Design, Manufacturing and Service with Big Data. *International Journal of Advanced Manufacturing Technology* 94 (2018), 3563–3576.

[66] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee. 2019. Digital Twin in Industry: State-of-the-Art. *IEEE Trans. on Industrial Informatics* 15, 4 (2019), 2405–2415.

[67] Y. C. Tay. 2018. *Analytical Performance Modeling for Computer Systems* (third ed.). Morgan & Claypool.

[68] B.D. Theelen. 2004. *Performance Modelling for System-Level Design.* Ph.D. Dissertation. Eindhoven University of Technology.

[69] B. D. Theelen, O. Florescu, M. Geilen, J. Huang, P. H. A. van der Putten, and J. Voeten. 2007. Software/Hardware Engineering with the Parallel Object-Oriented Specification Language. In *Proc. 5th ACM & IEEE Int. Conf. on Formal Methods and Models for Co-Design, MEMOCODE'07*. IEEE Computer Society, 139–148.

[70] F. W. Vaandrager. 2017. Model Learning. *Commun. ACM* 60, 2 (2017), 86–95.

[71] B. van der Sanden, J. Bastos, J. Voeten, M. Geilen, M. Reniers, T. Basten, J. Jacobs, and R. Schiffelers. 2016. Compositional Specification of Functionality and Timing of Manufacturing Systems. In *Proc. Forum on specification & Design Languages, FDL'16*. IEEE.

[72] B. van der Sanden, Y. Blankenstein, R. Schiffelers, and J. Voeten. 2021. LSAT: Specification and Analysis of Product Logistics in Flexible Manufacturing Systems. In *Proc. 2021 IEEE 17th Int. Conf. on Aut. Sc. and Eng., CASE'21*. IEEE.

[73] J. van Pinxten, U. Waqas, M. C. W. Geilen, T. Basten, and L. Somers. 2017. Online Scheduling of 2-Re-entrant Flexible Manufacturing Systems. *ACM Transactions on Embedded Computing Systems* 16, Article 160 (2017).

[74] M. Vierhauser, H. Marah, A. Garmendia, J. Cleland-Huang, and M.l Wimmer. 2021. Towards a Model-Integrated Runtime Monitoring Infrastructure for Cyber-Physical Systems. In *Proc. IEEE/ACM 43rd Int. Conf. on Software Engineering: New Ideas and Emerging Results, ICSE-NIER'21*. IEEE, 96–100.

[75] M. Voelter. 2014. *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages.* dslbook.org.

[76] J. Voeten. 2002. Performance Evaluation with Temporal Rewards. *Performance Evaluation* 50, 2-3 (2002), 189–218.

[77] W. Wolf. 2010. *High-Performance Embedded Computing: Architectures, Applications, and Methodologies.* Elsevier.

[78] K. Wolter and P. Reinecke. 2010. Performance and Security Tradeoff. In *Formal Methods for Quantitative Aspects of Programming Languages, SFM'10*. Springer, 135–167.

[79] L. D. Xu and L. Duan. 2019. Big Data for Cyber Physical Systems in Industry 4.0: a Survey. *Enterprise Information Systems* 13, 2 (2019), 148–169.

[80] N. Yang, K. Aslam, R. Schiffelers, L. Lensink, D. Hendriks, L. Cleophas, and A. Serebrenik. 2019. Improving Model Inference in Industry by Combining Active and Passive Learning. In *Proc. 2019 IEEE 26th Int. Conf. on Software Analysis, Evolution and Reengineering, SANER'19*. IEEE, 253–263.

[81] H. Zhang, Y. Shu, P. Cheng, and J. Chen. 2016. Privacy and Performance Trade-off in Cyber-Physical Systems. *IEEE Network* 30, 2 (2016), 62–66.