Design space exploration for distributed cyber-physical systems: state-of-the-art, challenges, and directions

Herget, M.; Saadatmand, F.; Bor, M.; Alonso, I.; Stefanov, T.P.; Akesson, B.; ... ; Skavhaug, A.

Design Space Exploration for Distributed Cyber-Physical Systems: State-of-the-art, Challenges, and Directions

Marius Herget*, Faezeh Sadat Saadatmand†, Martin Bor*,
Ignacio González Alonso‡, Todor Stefanov†, Benny Akesson*§, and Andy D. Pimentel*

* Informatics Institute (IvI), University of Amsterdam, Netherlands
† Leiden Institute of Advanced Computer Science, Leiden University, Netherlands
‡ ASML Netherlands B.V., Netherlands
§ ESI (TNO), Netherlands

2022 25th Euromicro Conference on Digital System Design (DSD) | 978-1-6654-7404-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSD57027.2022.00090

***Abstract*—Industrial Cyber-Physical Systems (CPS) are complex heterogeneous and distributed computing systems, typically integrating and interconnecting a large number of subsystems and containing a substantial number of hardware and software components. Producers of these distributed Cyber-Physical Systems (dCPS) face serious challenges with respect to designing the next generations of these machines and require proper support in making (early) design decisions to avoid expensive and time consuming oversights. This calls for efficient and scalable system-level Design Space Exploration (DSE) methods for dCPS.**

In this position paper, we review the current state of the art in DSE, and argue that efficient and scalable DSE technology for dCPS is more or less non-existing and constitutes a largely unchartered research area. Moreover, we identify several research challenges that need to be addressed and discuss possible directions for targeting such DSE technology for dCPS.

***Index Terms*—Distributed Cyber-Physical Systems, Design Space Exploration, Workload Modelling, Performance Modelling, Model Inference, Workload Dynamism**

I. Introduction

CYBER-PHYSICAL SYSTEMS comprise one of the largest information-technology sectors worldwide, driving innovation in other crucial industrial sectors, such as health industries, industrial automation, robotics, avionics and space. Nowadays, the embedded compute infrastructure of complex CPS is based on heterogeneous multi-core or many-core systems, which are distributed, and connected via complex networks [1]. Manufacturing companies of dCPS, such as ASML, Canon Production Printing, and Philips, face important challenges in designing their next-generation lithography scanner machines, industrial printers, and X-ray machines, respectively [16]. Typically, these machines are very complex dCPS that integrate and interconnect a large number of sub-systems containing multiple dependent compute nodes (hardware and software components) that perform different tasks, e.g., data processing, control, monitoring, thereby realising a wide range of functionality and features. Designers of such systems need quick answers to so-called "what-if" questions concerning design decisions and their impact on non-functional aspects, such as system performance, cost, or energy consumption. Hence, the academic community, along with the industrial sector, are calling for more research on efficient and scalable system-level DSE methods for dCPS [32]. These need to integrate appropriate application workload and system architectures models, simulation and optimisation techniques, as well as supporting tools, to facilitate the exploration of a wide range of design decisions.

In this position paper, we argue that such efficient and scalable DSE technology for dCPS is more or less non-existing and constitutes largely unchartered research area. Moreover, we identify several research challenges that need to be addressed and discuss possible solution directions when targeting DSE technology for dCPS. The four major contributions of this position paper are: (i) the presentation of an abstracted general workflow that can be used to structure and position work in the area of DSE (Sections II and III), (ii) the identification and description of open scientific challenges in the area of Design Space Exploration for dCPS (Section IV), (iii) a simple experiment with a state-of-the-art DSE tool to demonstrate some of the mentioned challenges (Section V), and (iv) a description of our ongoing research addressing the aforementioned challenges (Section VI).

II. Background

Design Space Exploration (DSE) is the process of discovering one or many design solutions that best satisfy defined design objectives given a space of tentative solutions called *design points*. Although this research field is often seen in the context of hardware design, there is no explicit definition of which kind of design is targeted. Hence, design points can consist of pure hardware, or software, or a combination of both. The complexity of the complete design space is based on the defined design choices and is defined by the cartesian product of all those choices (all possible combinations). In general terms, DSE is either a single or multi-objective optimi-

This work was supported by the NWO Mastering Complexity (MasCot) research program in partnership with TNO-ESI (project 17930) and by ASML Netherlands B.V.

2771-2508/22/$31.00 ©2022 IEEE
DOI 10.1109/DSD57027.2022.00090

sation problem where a cost function of all design objectives is maximised or minimised subject to several constraints. Classic design objectives include energy consumption, cost, reliability, throughput, or a combination thereof. DSE has a long history in mathematics (as an optimisation problem) and computer science research domains. Nonetheless, within the latter area, it has been mainly researched and utilised within the field of embedded Systems-on-a-Chip (SoC) design [12, 23].

The precise implementation of a DSE process is subject to various factors, e.g., the specific system/use-case, desired accuracy, abstraction level, restrictions, design objectives, and available computing resources for the actual DSE process. Nevertheless, each implementation and approach mostly share similar steps, which can be abstracted in a typical workflow outline shown in Fig. 1. This workflow concentrates on the evolutionary aspect of DSE, i.e. desiging a new generation of machines based on existing CPS. In more detail, DSE can generally be defined by four steps: (i) the preparation (description and creation) of the models based on existing systems (Fig. 1a), (ii) the construction of the design space (Fig. 1b), (iii) the systematic analysis of the set of all possible design points in the design space, including the evaluation of each individual design point based on specified criteria (Fig. 1c), and (iv) the further processing/presentation of the results (Fig. 1d).

The construction of design points is split up into two substeps: Modelling and Design Space Construction. Modelling (see Fig. 1a) is the preparation of the environment to be explored. This includes discovering and describing the system artefacts, like software applications, available/suitable architectural platforms, and design choices (e.g., various hardware options, scheduling of tasks to specific processors, or the system's topology). Complex and too detailed descriptions are often unsuitable for efficient, time-constrained analysis. Hence, an appropriate level of abstraction needs to be chosen for the models. The last substep is mapping all models into one abstract system representation. This should result in one designated model of combined software and hardware of the system to which the further DSE steps can be applied.

As seen in Fig. 1b, the constructed models can then be used to "span" a design space by creating all possible design points based on the design choices the models capture. A static/pre-exploration pruning can already be conducted based on external constraints or apparent incompatibilities within generated design points.

The third significant step of the general DSE workflow is the exploration of the design points, followed by the results of the DSE process. During exploration (shown in Fig. 1c), a search algorithm is used to determine which design points need evaluation. Each of these points is then evaluated, and the results are stored. An evaluation of a design point is defined as studying the extra-functional behaviour of system configurations (models), producing measurable KPIs to optimise for. Design points are dynamically pruned depending on the search strategy and information from newly obtained evaluation results. This can drastically decrease the size of the

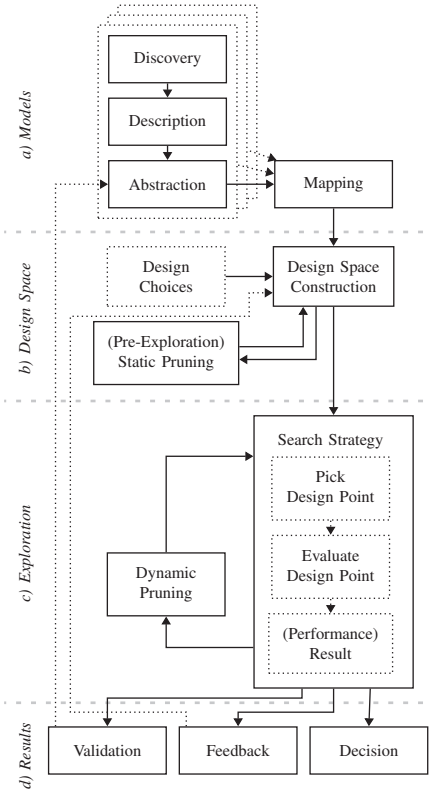

Fig. 1: General DSE workflow

design space. The exploration either yields intermediate results or a conclusive recommendation for design decisions (see Fig. 1d). Intermediate results can be used to tune the models, adjust the search algorithm via feedback loops, or validate either. Consequently, the Design Space Exploration workflow is able to be restructured as a multi-level, hierarchical approach where the abstraction level can be decreased in multiple runs. Nevertheless, in the end, recommendations for the design decisions are presented in a suitable form (e.g. as pareto fronts [19]) to the designer to guide decision making.

While the idea of this abstracted DSE workflow can be observed in various forms in different implementations; the explicit realisation is not limited to its specific arrangement. Hence steps can be skipped, exchanged or added depending on the use case. Nevertheless, this position paper will use this workflow as the structure for related work and open challenges.

III. Related Work

The goal of this section is to present major related work in form of general methods and techniques for specific steps during DSE, as well as related scientific studies in the field. This presentation is oriented to the abstract DSE workflow presented in Fig. 1 and focusses especially on *Models* and *Exploration* (Figs. 1a and 1c).

As already indicated in Section II, most implementations have common features in their DSE workflow. One key obser-

vation often used is the "separation of concerns" philosophy during modeling which, e.g., Kienhuis [15] realized as the *Y-Chart methodology*. Hereby, the functionality in the form of the application workload, the architecture of the underlying system, and the mapping between them are clearly separated. This idea has been adopted in many system modeling approaches. It is also indicated in Fig. 1a in the form of various models representing different concerns and their relationship expressed by a mapping layer. It has been also identified as an industry best practice for DSE by Van der Sanden et al. [32]. The initial models are – in the overwhelming majority of related work – created manually [20], but, with the increasing complexity of the systems, efforts have been made to infer them from recorded or discovered data [38].

In Fig. 1c, the evaluation of each design point, and the search strategy through the design space have been briefly mentioned. Pimentel [23] divides the evaluation of each design point in the design space roughly into three categories: (i) benchmark measurements of prototype implementations, (ii) simulation-based examination, and (iii) analytical model estimations. Analytical models allow for efficient evaluation of design points [23]. Mechanistic analytical models are built in a bottom-up fashion and capture the actual behaviour of the application and architectural elements by considering and accounting for a variety of events (e.g., cache misses, and resource contention) [13, 23]. In contrast to mechanistic models, empirical models, statistical inference, and machine learning techniques infer the performance from previously learned data. This "black box" approach requires less intimate knowledge of the mechanics of the modelled system and, therefore, is easier to develop.

The simulation-based approach mimics the system's behaviour and can be performed on various levels of abstraction. While a high abstraction level increases evaluation speed, details of the system are lost, and thus the accuracy is lower. Register-transfer level simulation is the lowest meaningful level for digital systems, whereby the digital signals between registers and combinational logic is explicitly mimicked. For each increasing level of abstraction there are various frameworks and implementations available [23] (e.g., cycle-accurate [5] or transaction-level modeling [6]). For exhaustive and complex systems, all these *program execution-driven* approaches are often still too cost-intensive and detailed for efficient evaluation of a large number of design points. Hence, the *trace-driven approach* focuses on collected application artefacts called *event traces*, reducing the modeling complexity [24]. While it potentially reduces the simulation cost significantly, an initial system configuration from which traces can be extracted has to exist already.

The second large sub-step of Fig. 1c are search strategies which are categorized as; (i) exhaustively evaluating every possible design point, (ii) random/intelligent sampling, or (iii) incorporating domain knowledge of the environment [12]. Panerati, Sciuto, and Beltrame [21] present an overview of how to choose the appropriate algorithm for the specific DSE use case by classifyng and comparing fifteen methodologies based on several metrics.

The final feature in Fig. 1c, orthogonal to the above-explained sub-step, is *static and dynamic pruning* of the design space. Gries [12] divides this into practical approaches: (i) hierarchical exploration, (ii) subsampling of the design space, and (iii) subdividing the design space. The hierarchical approach starts with a coarse model identifying promising design space regions, followed by a low-level, more detailed model exploration within each region. The subsampling approach chooses regions based on randomness or patterns. The third approach explores independent parts of the design space one by one and combines the resulting sub-solutions afterwards. In the end, all classes identify potential regions of design points for further exploration while effectively pruning the design space.

While the domain of DSE is an active topic in many research areas, within the computer science domain, it mostly evolves around Systems-on-a-Chip and Multiprocessor Systems-on-a-Chip (MPSoC) designs. For example, SESAME [22] is a trace-driven software framework for the latter category that allows the designer to do system-level modeling and simulation at a high level of abstraction with a Genetic Algorithm (GA) [8]. The framework follows the Y-Chart approach for trace-driven modeling and supports scheduling during mapping. A GA is used to search the design space [8], followed by a simulation-based evaluation of the chosen design points.

Within the SoC/MPSoC domain, many efforts study the allocation (spatial binding) of application tasks to (heterogeneous) processing elements. The temporal binding (scheduling) is often removed from the scope since it dramatically increases the design space. Nevertheless, some research has been conducted around scheduling and allocation in various environments [2, 3, 17, 26, 27, 28]. In 2022, Wan and Zeng [35] proposed a novel but highly limited co-design methodology specialized for modular CPS within a production setting.

In general, the domain of DSE for Cyber-Physical Systems (CPS) has slowy received more attention from the scientific community in recent years. While Mühleis et al. [18] mainly focused on DSE on electronic system level, Vanommeslaeghe et al. [33] incorporated domain knowledge to explore an electric DC motor. DISPATCH is a two-step methodology especially designed for CPS [30] that improved the sample efficiency for electrical circuit benchmarks. Nevertheless, all these explored CPS designs are relativly small arrangements.

The automotive sector is one of the leading forces for more complex systems; Canedo and Richter [7] presented a Functional Modeling Compiler approach for realistic automotive architectures (i.e., validate new Electronic Control Units and control strategies). Other literature includes DSE for controller area network (CAN) systems [36], or electrical/electronic (E/E) architecture component platforms for modern automotive systems [11].

While all authors explored different types of Cyber-Physical Systems, none explored widely distributed architectures outside of a specific subsystem. Zhang et al. [37] conducted an extensive gap analysis for state-of-the-art DSE methods in

the context of the automotive sector in 2017 and concluded that besides many other challenges, almost all of the DSE methods made oversimplifying assumptions for the vehicle's subsystems (i.e., ECUs, network topology).

In a more general sense, DSE for distributed (heterogeneous) computing systems is also a largely unchartered research area. In 2010, Fummi et al. [9] introduced a mathematical language to model distributed applications focussing on exploring communication infrastructures like wireless sensors or peer-to-peer networks. A similar but more practical approach has been made by Tanganelli et al. [29]. Their methodology looked at systems for smart environments with "Fog Computing". While this area has some similarities with our domain of industrial distributed Cyber-Physical Systems, their abstraction level is still too high (i.e., a fire detection and surveillance system for an office floor plan).

To the best of the authors' knowledge, very few to no studies have considered system-level DSE for distributed Cyber-Physical Systems, and there still exists a significant number of open scientific challenges in terms of scalability, performance, and accuracy of these DSE methodologies.

IV. Scientific Challenges

Designers of industrial distributed Cyber-Physical Systems (dCPS) face the challenge of highly complex systems that cannot be realistically understood in reasonable detail by one individual alone. Consequently, there is an enormous amount of subsystems with heterogeneous hardware and ample opportunities to modify each and every one of them. While designing new and advanced products, a considerable number of design objectives, highly complex models, an inconceivable number of design decisions, and a lot of internal and external factors have to be taken into account. This results in a vast design space with a potentially much larger number of design points that current methodologies cannot handle.

In the survey-based study of Van der Sanden et al. [32], industrial dCPS companies express great interest to research and develop efficient DSE methods to support the design process. One example is ASML, a leading force in the semiconductor industry, developing and manufacturing chip lithography machines. The improvement goals for these intricate dCPS include, but are not limited to, increasing the throughput of produced wafers, improving the quality and reliability of the final product, avoiding workflow disruptions, and decreasing the required maintenance. These machines consist of over 500 software processes distributed to tens of processors with hundreds of cores distributed over a network. Additionally, hundreds of specialised sensors and actuators within smaller subsystems are (partly) controlled by these software processors. Moreover, many more design decisions need to be taken into account: e.g., network topology, speed of links, the configuration of network components, or database infrastructure. Although many of these design points can be pruned early due to design constraints, the resulting design space is still immense.

This section is dedicated to identifying, describing and appraising (scientific) challenges based on observations during our research. It has to be mentioned that we limit the scope of this analysis to complex industrial, distributed Cyber-Physical Systems (e.g., ASML lithography machines or Canon industrial printers).

A. Modelling complex dCPS

The first step in having a comprehensive DSE is creating the models of the system at an appropriate level of abstraction. dCPS consist of multiple heterogeneous subsystems on a complex network with much interaction, usually running a combination of new and legacy software. Legacy software within the industrial sector can be multiple years, or even decades, old without detailed documentation. Hence, the creation of the models is, in comparison to SoC and MPSoC systems, often a huge challenge, and completely manual generation is not feasible in a timely manner. *(Semi-)automatic model inference* of the application and platform models is, therefore, the only viable option for DSE of dCPS.

A sufficiently accurate model mimics the system's behaviour by being a virtual "re-creation". Based on the separation-of-concerns idea, a comprehensive system representation consists at least of an application workload and a platform architecture model. As already depicted in Fig. 1a, the discovery and gathering of relevant information is the first step in describing a system (e.g., via doing static analysis as well as dynamic analysis or monitoring and capturing of event traces). A significant challenge during the abstraction of this description is handling conflicting requirements regarding evaluation speed, modeling effort and accuracy. Capturing the software behaviour of the dCPS in the models needs to be accurate enough to support trustworthy DSE. At the same time, it also must allow for fast (co-)simulation (and should thus be abstract enough). This calls for an extensive infrastructure that helps move the abstraction level dynamically to find the best balance to meet speed and accuracy requirements. This type of knowledge-based and automatic modeling is challenging.

Thus, one specific scientific challenge is the development of concepts and techniques to automatically infer the (initial) platform architecture and application workload models. This extends to the amalgamation of all models in application-to-resource mappings and scheduling.

B. Scalable Design Space Exploration (DSE)

The fundamental problems of a vast design space has been briefly discussed. The application workload (typically containing hundreds of software processes) and the various mappings of the application workload on these platforms already make the search space vast, but this is exacerbated by the fact that application workloads in dCPS typically are not static. For example, in the case of ASML lithography scanner machines, the application workload behaviour is highly dependent on factors such as the wafer size, recipe (mask) complexity, required accuracy, application configuration settings, external influences like customer or fab cronjobs, emergent dynamic

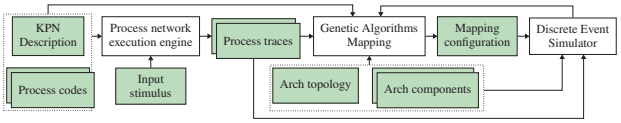

Fig. 2: SESAME workflow [10]

behaviour of the system, etc. All of these factors complicate the previous modeling efforts and contribute to an ever-increasing number of design points.

This calls for efficient and scalable search and pruning strategies. It is unclear if, and actually not to be expected that, state of the art in DSE (see Section III) is able to handle the much larger design spaces of dCPS. This is further exacerbated by the fact that design point evaluations take longer for dCPS compared to classical DSE of on-chip systems (SoCs and MPSoC) simply because of their complexity and, hence, allows for fewer evaluations in the same amount of time. Although the minimisation of the computational effort is primarily dictated by the chosen abstraction level for the models, there is also still limited but significant research potential in, e.g., optimisation of the simulation codebase (acceleration of each evaluation), exploitable parallelisation by designing algorithms with limited dependencies, or support for distributed parallel computing.

While the right search strategy highly depends on the design space and its characteristics, a general observation can be made; there is a trade-off between the effort required to configure an algorithm for a given design space, the quality of the results and the total number of design points evaluated. Similar observations apply to choosing a pruning technique; how many design points can be pruned before the accuracy of the DSE is too low, while the cost of implementation and execution is still feasible/balanced?

Consequently, another scientific challenge is the need for new search and pruning strategies and efficient design point evaluation, which all need to be sufficiently scalable. Since there most likely will not be a perfect solution, combining various concepts and algorithms is not out of the scope.

V. Experiments

The previous section presented and discussed open scientific challenges to be faced when developing the next generation of DSE for complex dCPS. In this section, some of these challenges are further motivated through a suite of simple DSE experiments.

The experiments are designed using the SESAME (Simulation of Embedded System Architectures for Multilevel Exploration) framework, which is a well-known system-level modeling, simulation, and exploration framework for embedded MPSoC systems introduced by Pimentel, Erbas, and Polstra [22]. Fig. 2 shows the various components of the SESAME modeling, simulation, and exploration framework. The application models (modeling a network of application processes and their interactions) are mapped onto architecture models via a transitional scheduling layer. A process network execution engine executes the application models generating application event traces for every process. These process traces are afterwards used to drive a transaction-level simulation model of the underlying platform architecture via a Discrete Event Simulation (DES). A Genetic Algorithm (GA) is used to drive the search within the design space, exploring and evaluating the fitness (using the DES) of different application-to-architecture mappings. More detailed descriptions and explanations can be found in [10, 22]. We selected SESAME for this experiment since it models MPSoCs and their application workload at a high level of abstraction, achieving high performance in simulating and exploring these MPSoCs.

The structure of our experiment is based on exploring the course of key performance metrics when scaling up a simple DSE example. As a building block for our application models, we picked a readily available (multi-media) application from the domain of streaming applications since, at a high level of abstraction, many industrial CPS can be modelled as streaming applications. Since these MPSoC applications are typically small in size (only six processes in our selected application), we duplicated our application a number of times to simulate larger workloads. As a basis for our platform model, we use an MPSoC architecture in which four different processor types can be deployed, each having a different cost and performance. For simplification, all CPUs are connected to and communicate via one bus and shared memory. To avoid the bus becoming a bottleneck, we mimic a system with ideal communication. This means that all memory and communication costs have been set to zero. To model an industrial dCPS system, we have scaled the MPSoC platform model so that it can contain up to a few hundred cores.

In our experiment, we explored the different mappings of the application workload (processes) onto the underlying platform architecture. We used a GA with a population size of 20, and each DSE run consists of 50 search iterations (generations) of the GA (i.e., each DSE run explores 1000 design points). The metrics studied are (i) runtime of the complete DSE, (ii) the convergence rate of the GA, and (iii) the amount of generated infeasible design points, but we will focus our discussion on the DSE runtime. Each experiment was conducted between two and four times with the same input parameters, averaging the results in the end. Due to limited time and computing resources some experiments have been performed only once or twice. These are indicated with a white shading in Fig. 3. All experiments were run on a computer with an Intel Xeon E5-2650v4 (12 cores) running at 2.2 GHz using 64GB of RAM, running Ubuntu 18.04 LTS.

Fig. 3 shows the results for the DSE mean runtime when varying the number of processes in the application workload and the number of processing cores in the underlying platform in the form of a heatmap. The 458 experiments had a combined runtime of roughly 64 days and 2 hours. On average, the results - within multiple runs of the same parameters - are within a 3.8% deviation respectively (with a maximum of 13.99%).

Our results show that the DSE runtimes significantly in-

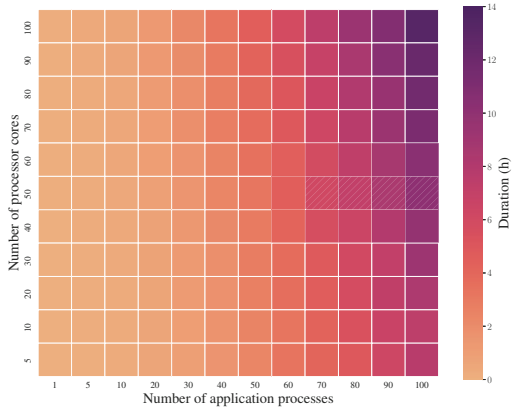

Fig. 3: Heatmap of the DSE runtime results when varying the number of application processes and number of cores

crease (superlinear) with the number of used application processes and processing cores due to the higher simulation overheads of the large(r) systems. This is especially true when scaling the number of application processes. To put it more simply, exploring a simplified design space with 100 applications and tens of cores is already a significant time investment and cannot be efficiently evaluated by a currently state-of-the-art framework. The previously described use case of ASML machines already has a considerably higher number of both parameters. In addition, these dCPS have a more complex network with more communication and increased dependencies. We would like to stress the fact that our simple experiment does produce an extremely large mapping space (i.e., different application processes to processor mappings). However, the design space itself is still rather "predictable". Evidently, this is due to the simplicity of the modelled platform (e.g., no distributed system but an idealized bus-based system, limited component diversity, etc.), as well as the simplicity of the modelled application workload (e.g., it does not contain any dependencies between the small, replicated applications). Considering this, we expect that the runtime of real-life use-cases will be even higher than these results.

Since we have made simplifying assumptions that, e.g., guarantee that every application process can always communicate with another process irrespective to which processing core it is mapped, our experiments do not show overhead for invalid design point repairs. However, it is expected to increase with a more complex and realistic use case significantly. Given the above, we expect our experiment to constitute a best-case scenario.

In conclusion, our experiments show that the current state-of-the-art methodologies are - in the best case - only partially capable of running DSE for dCPS. For more realistic use cases, we believe that a slower convergence behaviour and a less effective search process due to an increasing number of infeasible design points will significantly slow down the DSE process even further.

VI. Proposed Approach

Section IV presented the currently biggest scientific challenges in designing DSE for industrial, complex dCPS. Companies of these systems have a growing interest in facilitating DSE to support early design. Hence, in this section, we sketch a possible approach to address these challenges in the context of our collaboration with ASML. This discussion will mostly focus on the first challenge of automatically inferring the system models used in DSE. However, we want to remind the reader that this is a position paper and therefore does not present an actual implementation of the proposed solution. Moreover, similary to Fig. 1, we focus on the evolutionary aspect of DSE.

A. Modelling

For the models, we are following the well-known principle of "separation of concerns" [14]. Therefore, the (distributed) platform architecture model, the application workload model (i.e., the software processes executing on the platform architecture), and the mapping of the application processes/tasks to the platform resources are modelled separately. This approach allows for easy re-use of the different models involved. Consequently, it facilitates effective DSE of various platform architectures, application workload variants, and application-to-resource mappings.

Application Workload Model: We are following a pragmatic step-wise approach, starting with an initial definition and investigation of a first-order model and applying it on (part of) an ASML scanner machine software infrastructure to evaluate the ability of the model to capture the workload behaviour of such a complex real-world infrastructure accurately enough. Then, if needed, we will perform several model refinement steps to fine-tune the (first-order) model to increase or decrease its accuracy.

Our first-order model is based on the observation that a complex dCPS software infrastructure could be considered – at a high level – as hundreds of software processes triggered by events and exchanging messages among each other. Different processes perform different tasks, such as data processing, control, monitoring, logging/reporting, etc. Once a process is triggered, it performs a certain amount of computation and communication (this is called *firing*), and when ready, it *exits* (it stops and waits to be triggered again). Such fire-and-exit behaviour repeats during the whole life cycle of a process. Therefore, we model the workload behaviour of the software infrastructure as a directed graph defined by the tuple *(P, Ch)*. As depicted in Fig. 4, *P* is a set of processes ($P = \{p_0, p_1, \ldots, p_o\}$), where each process models/corresponds to a process of the dCPS software infrastructure. *Ch* is a set of communication channels ($Ch = \{ch_0, ch_1, \ldots, ch_k\}$) modeling the exchange of control messages (triggering events), status messages (process state), and data messages (chunks of data to be processed or stored).

ASML machines, as a dCPS, can have different modes of operation. Hence, we consider that each process consists of

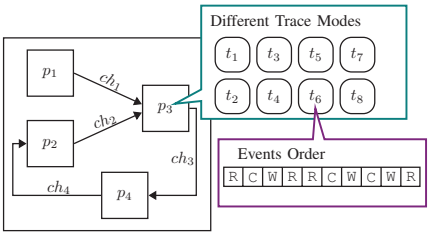

Fig. 4: Our first-ordered workload model

a set of traces T ($T = \{t_0, t_1, \ldots, t_m\}$). T allows capturing dynamically changing workload behaviour of a process since each trace captures, in an abstract way, the workload behaviour of a process in a specific system mode (configuration) during a single firing. Each trace is defined as a finite sequence of coarse-grained abstract events ($\{e_0, e_1, \ldots, e_n\}$), modeling the type and order of actions performed during a single firing. Event e can be *compute* (C), *read* (R) or *write* (W). The compute events imitate computational actions with an abstract workload described by the number of cycles needed to execute that computation. Events read and write mimic communication actions happening over communication channel *Ch* (e.g., memory accesses, communication between processors, or disk I/O depending on the mapping of *Ch*). Communications are described by the size (in terms of bytes) and type (control, data, or status) of a message they transfer besides the source and destination processes.

All in all, our first-order application workload model, adresses the challenges of system-level DSE of complex dCPS (see Section IV-A) with the following characteristics:

1) *The model is abstract and coarse-grained.* This potentially allows capturing the whole software infrastructure behaviour of a complex dCPS effectively at a high, system-level abstraction. The process traces can be (re)played, and their events synchronised relatively fast, thereby generating a proper workload for the platform model of the dCPS;
2) *The model is as timing and architecture agnostic as possible.* Avoiding timing and architecture artefacts in the model attributed to a specific hardware platform gives us very high flexibility to efficiently explore alternative mappings (alternative resource allocations, process bindings, and scheduling strategies) of the application workload onto a wide variety of existing/new hardware platforms;
3) *The model is dependency-aware.* Analysing the dependencies between processes, captured explicitly by the channels in the model and due to the exchange of messages of different types, allows us to effectively explore and exploit different degrees of parallelism when the application workload is mapped onto a platform;
4) *The model is mode-aware.* Capturing different modes by the set of traces in a process allows us to model different workload scenarios, thereby representing different system configurations and enabling workload dynamism modeling by switching between traces when generating workloads for the platform model.

Our first-order model does not follow the semantics of any well-known Model of Computation (MoC) [25]. Initially, our DSE should not be limited to the specific analysis and expressive power of a formal MoC to effectively capture the heterogeneous nature of the software infrastructure behaviour. As we explained before in Section IV-A, the software infrastructures of dCPS, such as ASML lithography machines, are too complex to be modelled manually. Hence, automated model inference is required. To infer our first-order model, we need runtime monitoring and data collection to obtain real software execution traces that typically contain time-stamped computation, communication, and synchronisation events with descriptors (e.g., the unique ID of a process, source/destination of a communicated message, etc.). The timing information in the traces depends on the platform architecture of the dCPS and the specific scheduling/mapping of the software infrastructure on this platform. Therefore, we need to transform this data in order to reduce time- and hardware-dependent information to extract the time and platform architecture agnostic workload model.

Architecture Platform Model: The platform architecture models represent the underlying system infrastructure of the dCPS. The inference of the model is initially based on the current architecture of the dCPS. Hence, we use a custom pipeline to discover, describe and abstract the available infrastructure:

1) *Discover network topology.* Firstly, starting at a main entry point, we record topology data by observing the network traffic and network interfaces. This includes, e.g., network connections, configurations of interfaces and network components (e.g., switches, routers, or hubs). Ethernet information like DNS/DHCP configurations, forwarding tables, etc., as well as data from custom connection techniques, are collected. This may or may not be manually supplemented with custom data from the designer. After that, all data are used to generate a network topology model.
2) *Computational components.* The first steps should have identified all components within the dCPS. Every component is addressed based on the network topology model, and hardware information (e.g., processor, memory, or storage) can be obtained.
3) *Description.* The information from the first two steps can then be combined by adding the discovered computational information (step 2) of the components to the network topology model (step 1). This results in a complete description of the used hardware.
4) *Abstraction.* The resulting network topology and components model is then abstracted to a discrete-time model. Key performance metrics are summarised into performance indices that describe each component's capabilities (e.g., a processing index, which describes how many computations can be executed per time unit). This abstraction is highly dynamic since it depends on the defined abstraction level, i.e., an index can describe the fundamental component (e.g., database write entries per

second) or specific characteristics (e.g., separate indices for processing speed and memory capabilities).

It has to be noted that depending on the specific use-case and dCPS custom discovery and description techniques may need to be implemented (e.g., for specific sensors or actuators).

Mapping: The mapping concept determines the binding of application processes (i.e., their event traces) to the modelled platform resource. The mapping model allocates the computational events to the platform resources by considering the mapping policy, hardware platform specification, inter-process communications, and several other constraints (i.e., access to specialized hardware like sensors). Since this step of allocation and scheduling already chooses specific parameters for design decisions, it is integrated into our approach's design space generation.

B. Scalable Design Space Exploration (DSE)

While generating the design space, we distinguish between *structural* and *behavioural* design decisions, resulting in two virtual design spaces. Structural design decisions change the models' structural integrity, e.g., modifying the network topology, creating or deleting communication channels between components, or exchanging key characteristics of a component (e.g., switching a GPU to an ASIC). The behavioural design decisions do not modify the structure of the system, but change the behaviour of the models, e.g., increasing or decreasing a performance index, or changing the mapping of specific processes to a different computing component.

Each dCPS design instance will be evaluated using a discrete-event simulation model of the workload model mapped onto the platform model using event traces. These simulations will be structured in *campaigns* [4, 34] – parallelising the evaluation of multiple independent design points simultaneously on a distributed computing cluster. While structural design decisions result in the need to rebuild a simulation model, behavioural decisions can be implemented as input parameters for simulation models. This allows for optimising the search strategy by creating and compiling a few design points in the structural design space with a range of parameters for behavioural design decisions. Additional techniques to deal with the vast design space can, for starters, be borrowed (and then extended) from the state-of-the-art in MPSoC DSE, such as hierarchical DSE methods (see [23]), and design space pruning by adding domain knowledge to the search algorithm (see [31]).

VII. Conclusion

In this position paper, we argued for the need for efficient and scalable Design Space Exploration (DSE) technology for distributed Cyber-Physical Systems (dCPS). Although there already exists a large body of research on DSE of SoC and MPSoC designs, there are no comprehensive research efforts to integrate DSE for the much more complex dCPS. Hence, we presented some of the major (scientific) challenges for this endeavour, namely the automatic generation of sufficiently accurate (application and platform) models and the need for efficient and scalable exploration techniques. To support some of our claims, we have demonstrated, using a simple experiment, that the DSE runtime, even for a best-case scenario, is significantly increasing with the number of used application processes and processing cores. Finally, we positioned our ongoing research in light of these challenges. In particular, the development of a workload and platform model that can be automatically inferred from process traces of industrial dCPS, as well as efforts to optimise the exploration of the vast design spaces in the domain of dCPS.

References

[1] Benny Akesson et al. "A comprehensive survey of industry practice in real-time systems". In: *Real-Time Systems* (Nov. 11, 2021). DOI: 10.1007/s11241-021-09376-1.

[2] Benny Akesson et al. "An efficient configuration methodology for time-division multiplexed single resources". In: *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. 2015. DOI: 10.1109/RTAS.2015.7108439.

[3] Sudarshan Banerjee et al. "Design space exploration of real-time multi-media MPSoCs with heterogeneous scheduling policies". In: *Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '06)*. Oct. 2006. DOI: 10.1145/1176254.1176261.

[4] Pablo Andrés Barbecho Bautista et al. "Large-Scale Simulations Manager Tool for OMNeT++: Expediting Simulations and Post-Processing Analysis". In: *IEEE Access* 8 (2020). DOI: 10.1109/ACCESS.2020.3020745.

[5] Anastasiia Butko et al. "Accuracy evaluation of GEM5 simulator system". In: *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. July 2012. DOI: 10.1109/ReCoSoC.2012.6322869.

[6] Lukai Cai and Daniel Gajski. "Transaction Level Modeling: An Overview". In: *Proceedings of the 1st IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. CODES+ISSS '03. Newport Beach, CA, USA: Association for Computing Machinery, 2003. DOI: 10.1145/944645.944651.

[7] Arquimedes Canedo and Jan H. Richter. "Architectural Design Space Exploration of Cyber-physical Systems Using the Functional Modeling Compiler". In: *Procedia CIRP* 21 (2014). DOI: 10.1016/j.procir.2014.03.183.

[8] C. Erbas, S. Cerav-Erbas, and A.D. Pimentel. "Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design". In: *IEEE Transactions on Evolutionary Computation* 10.3 (June 2006). DOI: 10.1109/TEVC.2005.860766.

[9] F. Fummi et al. "Modeling of communication infrastructure for design-space exploration". In: *2010 Forum on Specification & Design Languages (FDL 2010)*. Southampton, UK: IET, 2010. DOI: 10.1049/ic.2010.0135.

[10] Andres Goens et al. "Why Comparing System-Level MPSoC Mapping Approaches is Difficult: A Case Study". In: *IEEE 10th Int. Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*. Lyon, France: IEEE, Sept. 2016. DOI: 10.1109/MCSoC.2016.48.

[11] Sebastian Graf et al. "Multi-Variant-Based Design Space Exploration for Automotive Embedded Systems". In: *Proceedings of the Conference on Design, Automation & Test in Europe*. DATE '14. Dresden, Germany: European Design and Automation Association, 2014.

[12] M Gries. "Methods for evaluating and covering the design space during early design development". In: *Integration, the VLSI Journal* 38.2 (Dec. 2004). DOI: 10.1016/S0167-9260(04)00032-X.

[13] Rik Jongerius et al. "Analytic processor model for fast design-space exploration". In: *2015 33rd IEEE International Conference on Computer Design (ICCD)*. Oct. 2015. DOI: 10.1109/ICCD.2015.7357136.

[14] K. Keutzer et al. "System-level design: orthogonalization of concerns and platform-based design". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19.12 (Dec. 2000). DOI: 10.1109/43.898830.

[15] Albert Carl Jan Kienhuis. “Design space exploration of stream-based dataflow architectures”. PhD thesis. Delft, 1999.

[16] Brit Meier et al. *HTSM Systems Engineering Roadmap*. Tech. rep. 2020.

[17] Anna Minaeva et al. “Control Performance Optimization for Application Integration on Automotive Architectures”. In: *IEEE Transactions on Computers* 70.7 (July 1, 2021). DOI: 10.1109/TC.2020.3003083.

[18] Nina Mühleis et al. “A co-simulation approach for control performance analysis during design space exploration of cyber-physical systems”. In: *ACM SIGBED Review* 8.2 (June 2011). DOI: 10.1145/2000367.2000372.

[19] P. Ngatchou, A. Zarei, and A. El-Sharkawi. “Pareto Multi Objective Optimization”. In: *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*. Nov. 2005. DOI: 10.1109/ISAP.2005.1599245.

[20] Marcio F. S. Oliveira et al. “Model driven engineering for MPSOC design space exploration”. In: *Proceedings of the 20th annual conference on Integrated circuits and systems design*. SBCCI ’07. New York, NY, USA: Association for Computing Machinery, Sept. 3, 2007. DOI: 10.1145/1284480.1284509.

[21] Jacopo Panerati, Donatella Sciuto, and Giovanni Beltrame. “Optimization Strategies in Design Space Exploration”. In: *Handbook of Hardware/Software Codesign*. Ed. by Soonhoi Ha and Jürgen Teich. Dordrecht: Springer Netherlands, 2016. DOI: 10.1007/978-94-017-7358-4_7-1.

[22] A.D. Pimentel, C. Erbas, and S. Polstra. “A systematic approach to exploring embedded system architectures at multiple abstraction levels”. In: *IEEE Transactions on Computers* 55.2 (Feb. 2006). DOI: 10.1109/TC.2006.16.

[23] Andy D. Pimentel. “Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration”. In: *IEEE Design & Test* 34.1 (Feb. 2017). DOI: 10.1109/MDAT.2016.2626445.

[24] G S Sangeetha et al. “Trace-Driven Simulation and Design Space Exploration of Network-on-Chip Topologies on FPGA”. In: *8th Int. Symposium on Embedded Computing and System Design (ISED)*. Dec. 2018. DOI: 10.1109/ISED.2018.8703884.

[25] John E. Savage. *Models of Computation*. June 1998.

[26] Mario Schölzel and Peter Bachmann. “DESCOMP: A New Design Space Exploration Approach”. In: *Systems Aspects in Organic and Pervasive Computing - ARCS 2005*. Ed. by Michael Beigl and Paul Lukowicz. Berlin, Heidelberg: Springer, 2005. DOI: 10.1007/978-3-540-31967-2_13.

[27] Anirban Sengupta and Reza Sedaghat. “Integrated scheduling, allocation and binding in High Level Synthesis using multi structure genetic algorithm based design space exploration”. In: *2011 12th International Symposium on Quality Electronic Design*. Mar. 2011. DOI: 10.1109/ISQED.2011.5770772.

[28] Anirban Sengupta, Reza Sedaghat, and Pallabi Sarkar. “A multi structure genetic algorithm for integrated design space exploration of scheduling and allocation in high level synthesis for DSP kernels”. In: *Swarm and Evolutionary Computation* 7 (Dec. 2012). DOI: 10.1016/j.swevo.2012.06.003.

[29] Giacomo Tanganelli et al. “A methodology for the design and deployment of distributed cyber–physical systems for smart environments”. In: *Future Generation Computer Systems* 109 (Aug. 2020). DOI: 10.1016/j.future.2020.02.047.

[30] Prerit Terway, Kenza Hamidouche, and Niraj K. Jha. “DISPATCH: Design Space Exploration of Cyber-Physical Systems”. In: *CoRR* abs/2009.10214 (2020). arXiv: 2009.10214.

[31] Mark Thompson and Andy D. Pimentel. “Exploiting domain knowledge in system-level MPSoC design space exploration”. In: *Journal of Systems Architecture* 59.7 (Aug. 2013). DOI: 10.1016/j.sysarc.2013.05.023.

[32] Bram Van der Sanden et al. “Model-Driven System-Performance Engineering for Cyber-Physical Systems : Industry Session Paper”. In: *2021 International Conference on Embedded Software (EMSOFT)*. Oct. 2021.

[33] Yon Vanommeslaeghe et al. “Leveraging Domain Knowledge for the Efficient Design-Space Exploration of Advanced Cyber-Physical Systems”. In: *22nd Euromicro Conference on Digital System Design (DSD)*. 2019. DOI: 10.1109/DSD.2019.00058.

[34] Andras Varga. “A Practical Introduction to the OMNeT++ Simulation Framework”. In: *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*. Ed. by Antonio Virdis and Michael Kirsche. Cham: Springer International Publishing, 2019. DOI: 10.1007/978-3-030-12842-5_1.

[35] Guangxi Wan and Peng Zeng. “Codesign of Architecture, Control, and Scheduling of Modular Cyber-Physical Production Systems for Design Space Exploration”. In: *IEEE Transactions on Industrial Informatics* 18.4 (Apr. 2022). DOI: 10.1109/TII.2021.3097761.

[36] Yong Xie et al. “Security/Timing-Aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems”. In: *IEEE Transactions on Industrial Informatics* 15.2 (Feb. 2019). DOI: 10.1109/TII.2018.2851939.

[37] Xinhai Zhang et al. “Architecture exploration for distributed embedded systems: a gap analysis in automotive domain”. In: *2017 12th IEEE International Symposium on Industrial Embedded Systems (SIES)*. Toulouse, France: IEEE, June 2017. DOI: 10.1109/SIES.2017.7993377.

[38] Hao Zheng et al. “Model Synthesis for Communication Traces of System Designs”. In: *2021 IEEE 39th International Conference on Computer Design (ICCD)*. Oct. 2021. DOI: 10.1109/ICCD53106.2021.00082.