Symbolic Reasoning for Early Decision-Making in Model-Based Systems Engineering

Johan Cederbladh*,†, Loek Cleophas‡,§, Eduard Kamburjan¶, Lucas Lima‖**, Hans Vangheluwe**
*Mälardalen University, Västerås, Sweden – johan.cederbladh@mdu.se
†Volvo Construction Equipment, Eskilstuna, Sweden
‡Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands – l.g.w.a.cleophas@tue.nl
§Stellenbosch University, Stellenbosch, South Africa
¶University of Oslo, Oslo, Norway – eduard@ifi.uio.no
‖Universidade Federal Rural de Pernambuco, Recife, Brazil – lucas.albertins@ufrpe.br
**University of Antwerp and Flanders Make, Antwerp, Belgium – hans.vangheluwe@uantwerpen.be

***Abstract*—Hardware-intensive engineering domains are undergoing a paradigm shift toward digitalization. This is caused by increased technological advances in conjunction with stricter regulations on sustainability, while customer satisfaction and market competitiveness need to be achieved. In domains such as construction equipment and railway, strong foundations of Systems Engineering exist for development and management of products. With the advent of digitalization, Model-based Systems Engineering (MBSE) is increasingly seeing interest and industrial adoption. An expected benefit of an MBSE approach is the analysis capabilities early on in systems development. This is enabled by full-system models that can assist in early-stage design decision-making. This paper discusses how symbolic reasoning may facilitate knowledge reuse and support automation of design decisions based on previous development efforts and experiments. We situate this contribution in a typical industrial MBSE process and highlight its potential use and implementation.**

***Index Terms*—MBSE, Early phase, Symbolic reasoning**

I. Introduction

The traditional Systems Engineering (SE) development with islands of development centred around specific engineering domains and supporting tools is proving to not scale well. With increasing digitalization, companies and businesses look towards seamless integration of processes and tools. In particular, the notion of the digital thread in industrial landscapes is seeing growing interest [15]. A more interconnected development process offers many opportunities for stronger collaboration, traceability, and more widespread knowledge reuse. The traditionally document-centric development is less and less effective in dealing with the increasing complexity of systems and processes. In SE particularly, the increasingly applied paradigm of Model-based Systems Engineering (MBSE) fits well with the needs and incentives of increasingly digitalized workflows. The International Council on Systems Engineering (INCOSE) defines MBSE as "*[...] the formalised application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life-cycle phases.*" [18]. Its use is reported to have significant benefits in industry due to increased traceability, communication, and reuse [11], [13], [16].

Standard SE knowledge tells us that costs of correcting design or requirement issues increase exponentially as one progresses through the system development lifecycle [9], [18]. To take *correct* design decisions *earlier* is valuable and can reduce costs significantly. Using models in the early stages enables more sophisticated analysis and evaluation of system requirements and design. However, models in *early* phases of development inherently contain uncertainty due to the nature of the development progression [3]. Customer input and developmental progress introduce continuous shifts in the system to be delivered, while models at early phases often are semi-formal or highly abstract. Knowledge from previous projects or from previous development iterations may, however, be reused to increase decision capability.

Here, we describe how knowledge graphs and symbolic reasoning can improve analytical capabilities during the early phases of development. With knowledge graphs representing knowledge from previously performed experiments[1], symbolic reasoning can be used to analyse systems by reusing the results of experiments of *similar enough* nature. In particular, the use of an ontology for MBSE experimentation enables a user to launch queries that can automatically determine whether similar experiments have been performed in the past, and return an answer based on the recorded results of these earlier experiments, or suggest a new experiment to perform to provide such a response and thus also to expand the knowledge graph further.

The rest of the paper is structured as follows: Section II details the necessary background and industry motivation. Section III briefly discusses some related work. We detail a requirements analysis of our considered system in Section IV. In Section V, we detail and discuss an initial case study. Section VI concludes the paper.

This work was partly funded by the AIDOaRt project, an ECSEL Joint Undertaking (JU) under grant agreement No. 101007350, and the PeTWIN (294600) and SIRIUS (237898) projects funded by the RCN.

[1]We do not commit to a specific formal notion of an experiment here and use the term in a broad sense.

II. Industry background and motivation

MBSE promises to improve developmental efficiency and reduce the time to market of products. Industrial practice shows that there are many benefits to moving from traditional, document-centric to model-centric development, in the form of traceability, management of information, and clarity provided by diagrammatic views [5], [7]. More recently, MBSE has been employed to achieve a robust *digital thread* across the development stages of a typical process such as the common V-model [15]. In particular, with the advent of increased digitalization, increased collaboration can be expected due to a more integrated engineering landscape. By leveraging models and other digital artefacts, existing silos of development and operation processes can be joined to increase collaboration and knowledge reuse as well as to improve analysis capabilities.

One such capability is using *early validation* to analyze system viability in the design process quicker. Early validation is seeing increasing interest in MBSE [3], [6], and actors such as INCOSE predict an increasing amount of analysis using high-fidelity models in MBSE during design[2]. In particular, this includes the notion that partially or fully automated, machine-assisted operations could increase design effectiveness while reducing traditionally labour-intensive manual tasks [13]. Increased effectiveness of processes and tasks in MBSE directly affects the time-to-market, hinting at the potential value. As a system is developed, the *confidence* in the system progressively increases. MBSE adds value to the traditional SE process by introducing models as the main developmental artefacts instead of documents. Similarly, the early validation concept promises further value by extending the analytical capabilities via various methods.

In large complex systems such as heavy machinery, development is often performed in long development cycles, over many years. Additionally, most machines are part of product families or lines, with many points of variability in design and potential optional or configurable features [4]. Early validation can greatly impact decision-making and eventual system delivery time in this context. A primary need in early decision-making is to make "good enough" estimates of what design to pursue and, often more importantly, what not to investigate further. As such, it is a heuristic which supports pruning the search space. Such analysis is often based on expert engineering knowledge. Implementing automation in this decision-making process reduces manual labour and allows more widespread design validation by reducing the need for costly human expert intervention at each step.

In this context of machine-enhanced analysis capabilities to allow early decision-making, we propose our work on symbolic reasoning over knowledge of experiments represented by knowledge graphs.

III. Related work

As discussed above, the reuse of prior experiments by means of a knowledge graph can be seen as integrating a digital thread into a system. Indeed, our approach is closely related to the twinning paradigm (including Digital Shadows, Digital Twins (DTs) and digital threads) that has recently come to the fore. It uses digital counterparts to analyse and optimize real systems or performs what-if scenarios before new product variants are developed. Abadi et al. [2] consider decision-making for cyber-physical production systems with DTs. Our approach is not restricted to such systems nor to systems with DTs, but rather focuses on early-stage decision-making in which the system of interest and DT counterpart may not yet be realized. In addition, we separate concerns, modularizing knowledge graphs, reasoning, and experiment management. Cognitive Twins, building on DTs, have been proposed to support decision-making [12] in settings where DTs are available or make sense for a system of interest. D'Amico et al. [8] provide a recent overview of their use for improving maintenance management. Kamburjan et al. [10] similarly integrate knowledge graphs for self-adaptive DTs to make information about the DT available. In this work, they do not consider prior experiments, only current physical structure. In contrast to these works, we do not aim to limit the use of knowledge graphs to the runtime of the deployed DTs but provide a general framework for reasoning and reuse.

Beyond DTs, combining logical inference in rule-based expert systems with quantitative analysis (numerical simulation) was explored by Vangheluwe and Vansteenkiste [17], who combined high-level domain knowledge, in the form of rules, with low-level, case-specific knowledge in the form of simulation models, including initial values and parameters.

IV. Requirements Analysis

To enable early validation of system designs and reuse of the results and insights gained from these analyses in later development, one needs to support experiment design automation, as well as storage and management of, and *reasoning* about performed experiments. In the following, we give a more detailed analysis of the requirements for a framework that implements such operations.

To systematically enable experiment (and experiment result) management, a *uniform* and *formal* way to describe experiments is required. Ontologies have not only proven to be a suitable formalism to describe static structures, but they also offer reasoning, querying, and storage mechanisms. As we shall see below, we do require additional mechanisms beyond what ontologies offer. As a start, we formalize experiments by a suitable ontology and store them in a knowledge graph.

More precisely, the knowledge graphs must contain triples of the form $\langle \mathcal{Q}, \mathcal{E}, \mathcal{A} \rangle$ where $\mathcal{Q}$ is a question asked by the end-user of the framework, $\mathcal{E}$ is an experiment performed to answer that question, and $\mathcal{A}$ is the answer to question $\mathcal{Q}$, which is obtained by performing $\mathcal{E}$. We take $\mathcal{E}$ as a broad term, which can be virtual (e.g., a simulation) or real (e.g., a test scenario executed in the real world).

The framework, thus, consists of some interface that takes a new question q, decides on which experiments to answer that question, and derives an answer from the experiments. It needs

[2]https://www.incose.org/2023_redesign/publications/se-vision-2035

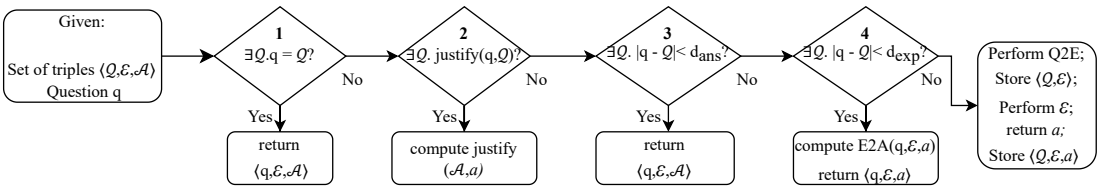

Fig. 1. Workflow of reasoning in *Q2A* before performing *Q2E*.

to perform two kinds of –preferably automated– reasoning about experiments:

- *Q2A* The first kind of reasoning goes from question q to answer a directly: Given a new question q, it tries to find some already performed and stored experiment $\mathcal{E}$, so that we can reuse its answer $\mathcal{A}$ to produce a.
- *Q2E* The second kind of reasoning goes from question to (a set of) experiments: Given a new question q, it tries to design some new experiment e that can be used to produce a new answer a.

Let us examine these reasoning components in more detail. First, we observe that *Q2A* implies *case-based reasoning* (CBR) [1], as it tries to produce new knowledge based on a set of previously solved cases and finding (and possibly adapting) similar ones. Second, *Q2A* implies both *deductive reasoning* and *planning*: based on general knowledge (about the domain, experiments, etc.), it deduces what experiment is most suited. As experiments are rarely monolithic tasks, their execution must be planned based on a decomposition into smaller tasks.

Fig. 1 shows a possible workflow for the overall system—covering *Q2A* which possibly invokes *Q2E*—, where a new question q is asked. We illustrate a simple CBR approach in the setting of an experiment manager, with 2 CBR systems; both implement the usual CBR cycle [1].

A pre-processing step (**1**) checks whether the exact same question has been asked before (and it and its answer have been stored). If this is the case, CBR is trivial, and the already available answer is returned. This is simple *instance-based reasoning* (or *memoization*). The next step (**2**) performs *symbolic* reasoning, where an answer $\mathcal{A}$ to an old question $\mathcal{Q}$ can be used to infer an answer a to a new question q based on some knowledge of the domain. As an example, consider the new question of whether some Construction Equipment (CE) machinery comes to a halt after d_{new} meters under certain conditions (often encoded in the form of parameters such as mass, slipperiness of the road surface, etc.). If there exists an old question that asks whether it comes to a halt after d_{old} meters with $d_{new} < d_{old}$, under the same conditions, with a recorded negative answer, then q can directly be answered negatively as well. This is based on our knowledge of continuity in the physical process of braking. Furthermore, if the new question has $d_{old} > d_{new}$, but with $|d_{old} - d_{new}| < \delta$ for some sufficiently small value δ, then it is close enough to the old question, making a new experiment unnecessary.

The next steps use a domain and application-specific distance measure $|\cdot|$ between *questions* to reuse either experiments or answers. *Distance-based* CBR is performed in two steps. Given some distance threshold d_{ans}, one tries to find a question that is not the same, but similar enough, to the new question to justify using the found question's answer as a new answer directly (**3**). Note that contrary to (**2**), where a justification step must be taken, here the answer is returned directly. Given some larger distance threshold d_{exp}, one tries to find a question whose answer does not justify a new answer, but whose experiment can be used to compute a new answer, taking into account the differences between $\mathcal{Q}$ and q (**4**).

In the CBR for **2** and **3**, different prior instances are *retrieved* to be *reused* (in terms of the CBR cycle). We thus consider them as two conceptually different cases. This also highlights the need for an appropriate *distance* measure.

Note that in cases **2**, **3**, and **4**, a new triple $\langle q, \mathcal{E}, a\rangle$ with new answer and question, but old experiment is stored. The new answer is justified differently: in the case of symbolic reasoning, the justification of a is from $\mathcal{A}$, $\mathcal{Q}$ and q, in the case of distance reasoning, one must also consider $\mathcal{E}$. The knowledge graph can also be used to store this information as a sort of provenance for the answer, detailing its justification beyond the relevant experiment.

If no answer can be inferred from previous experiments, it implies that we need to construct a new experiment to answer the question q, using the reasoning component *Q2E*. An experiment should be designed. Reasoning may help the experiment designer build a new experiment (both its workflow and its architecture). The workflow details the steps and the order in which they should be performed, while the architecture details the needed structures and their connection. Based on the type of question being asked, we can store in the knowledge graph information about the relations of workflows and goals to be achieved in experiments. If the goal of the question can be mapped to a workflow goal, we could provide that workflow for the engineer. Regarding the architecture, for instance, if the experiment is a simulation using Simulink®, according to the initial input and expected output, we can employ *deductive reasoning* to suggest which blocks can be used and how they should be connected. Nevertheless, the experiment design returned is a suggestion, which the engineer must still validate. However, suppose the engineer adjusts the experiment to fit their needs. In that case, the newly validated experiment can be stored in the knowledge graph to be used as a reference for future similar questions.

When the experiment is ready, it can be executed, and its result is returned as an answer. This result is also stored in the

knowledge graph for future reuse. Given that the elements are properly typed in the ontology, we can also reason over those types to provide knowledge on their structure, operations that can be performed on them, and also enhanced traceability. For instance, we may discover all the elements of a product family given a type that relates to a particular family.

V. Case Study

We illustrate our paper using an example from the previously mentioned CE domain. A crucial safety requirement for moving machinery is the maximum braking distance. CE machines travel at relatively low speeds but are typically heavy and many have large carry capacities. CE sites often contain many slopes with poor grip, with surfaces such as mud or gravel. Therefore it is important to understand the brake performance of CE machinery so that regulatory requirements ensuring safety are met. We display the case in Figure 2.

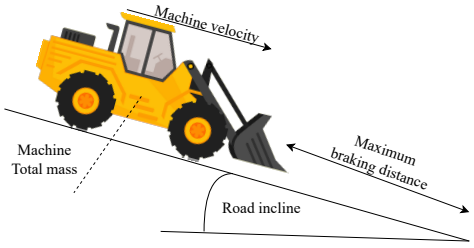

Fig. 2. Braking distance use case for the CE domain

The braking distance is a common function for all moving machines and a valuable target for reducing manual activities, particularly in early design phases. Many machines are developed in product families, and new products are often redesigns or improvements of existing machines, more often than not, in production. As such, there are many similar cases to extract knowledge from and gain valuable insight for decision-making. In some cases, such as the braking distance, the properties of interest can be reasonably reduced to a small number. The machine velocity is often set to a maximum. Likewise, the road type is often set for a particular condition (i.e., mud or gravel). This way, the machine's total mass (machine + load) and the road condition become the two most significant parameters, specifying conditions under which to verify braking capability. Such experiments, whether virtual or real, are performed for detailed prototyping. This historical knowledge can help guide developers when evaluating/testing *similar enough* designs. For example, Figure 3 highlights five tests for a brake system of a given machine with maximum operational speed. In this example, we consider two variables, *road incline* and *mass*, to visualize how symbolic reasoning works. However, the approach is not restricted to that.[3]

[3]The overall problem can be seen as a classification problem, where the experiments are the data points used to learn the classifier then used in *Q2A* step (**2**). Due to the cost of an experiment, the engineer is interested in minimizing the number of data points while also minimizing the uncertainty around the decision boundary, i.e., to perform active learning [14]. We leave this avenue open for future investigation as we do not focus on *Q2E* or the general workflow.

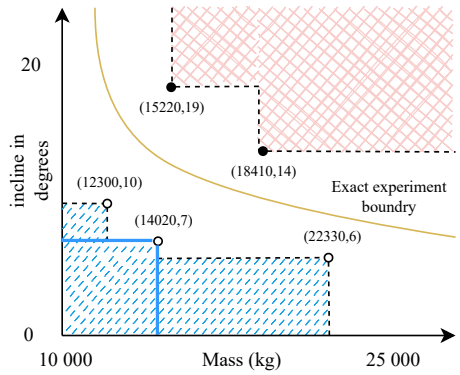

Fig. 3. Experiment map leveraging symbolic reasoning

We visualise the results in a map with open dots corresponding to passed tests and filled black dots corresponding to failed tests. Using our symbolic reasoning, we infer that the regions in blue (dashed) and red (crossed) can be considered passed and failed based on domain knowledge, which in this case corresponds to the laws of physics. For instance, if we know that a particular machinery weighing 15220 kg and a road incline of 19 degrees could not brake and stop as expected, for any value of mass greater than this, it will not come to a stop in time either. Also, for a greater incline angle, it will be more difficult to halt, given the force of gravity. In the dual case, if we know that a machine weighing 22330 kg and a road incline angle of 6 degrees was able to brake, any machine with less weight on a less inclined road will also be able to brake. Therefore, given the previously performed experiments, we can extend the points to areas. For any point inside these areas, no extra experiments are needed. Finally, in the observed case, there is some exact boundary between pass and fail, and depending on the aim of experiments, it could be interesting to identify it as part of a larger process of iterative experimentation. This example provides an intuition of the strategy, but other types of knowledge could be considered, for example, from the available models (e.g., system architecture, experiment workflows, and domain artefacts in general).

We can still apply some reasoning for the points in the blank area, given that domain-specific information on distances is provided in the knowledge graph. For instance, assume that we can replicate positive results for machinery with masses below 15000 kg and incline below 10 degrees to mass variability of less than 0.5%. Although a question for a machine with a mass of 14500 kg and an incline of 7 degrees is in the blank area of the graph, a positive answer can be inferred, thanks to the data of a previous experiment where the machine could brake and stop having a mass of 14020 kg and an inclination degree of 7 ($14500 < 15000$ and $|14500 - 14020|/14020 < 0.5\%$).

Finally, if these reasoning mechanisms cannot infer a valid result, a new experiment must be performed to provide an answer. Nevertheless, we can still support the engineer by $Q2E$ reasoning. First, if an experiment has already been executed to achieve the goals of the question, then, querying

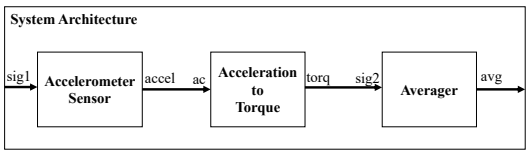

Fig. 4. Example of experiment architecture.

the knowledge graph, we can provide the workflow and architecture that has been used earlier, for the engineer to redo that experiment with different parameters.

Even in the case of no previous architecture, we can also support the engineer if information about structural elements is available in the knowledge graph. Consider that a simulation experiment needs to evaluate the average torque of a drivetrain given the acceleration sensor readings. Assuming that our knowledge graph stores information about block signatures, i.e., their input and output types, we can apply deductive reasoning to propose an architecture of blocks. Our reasoner carries out a search resulting in a proposed architecture connecting blocks, where the system output is torque average, and the system input is accelerometer sensor readings. A possible result is the one shown in Figure 4. It is an architecture composed of three blocks: one for the *Accelerometer Sensor*, which outputs acceleration from an input signal such as mechnical rotation; followed by a block that converts acceleration to torque; and the last block calculating an average of an input signal over time. The engineer must still evaluate whether the proposed architecture is suitable. Eventually, the final architecture must be stored in the knowledge base as an experiment e to the question q. After performing this experiment, the triple $\langle q, e, a \rangle$ is stored to use in future reasoning.

Hence, using these kinds of reasoning, an engineer could get a valuable feedback of a coarse-grained nature in *early* decisions. More detailed analysis is required for proper evaluation, certification, and eventual release, but this approach helps the developer to speed up development.

VI. Conclusion

This paper motivates the need for managing, and reasoning over, experiments to enable early validation and reuse of results and knowledge accumulated during the design of systems. Based on industrial experience, we have derived requirements for the formalization of experiments, particularly the reasoning components needed to realize our vision. In the paper, we use a SE case to describe the applicability of our approach based on this experience. A foreseen limitation of this approach is the data gathering, as it requires involved parties to store vast amounts of sensitive data, which might be challenging to gather. Similarly, the approach relies on cases where it is possible to reason about the similarity of experiments, which might require extensive domain knowledge. Especially the notion of what can be considered *close enough* results or experiments requires further investigation into how generalization might be done. Nonetheless, we believe the approach can increase process value and are now developing and implementing an architecture for reasoning about experiments, including an ontology for justifying answers to questions through experiments or reasoning.

References

[1] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, pp. 39–59, Mar. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=196108.196115

[2] M. Abadi, A. Chaimae, A. Asmae, and B.-A. Hussain, "A smart decision making system for the selection of production parameters using digital twin and ontologies," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 2, 2022. [Online]. Available: https://dx.doi.org/10.14569/IJACSA.2022.0130248

[3] K. Abdo, J. Broehan, and F. Thielecke, "A seamless and end-to-end approach for early and continuous validation of next-generation avionics platforms," in *Software Engineering 2023 Workshops*, 2023. [Online]. Available: https://doi.org/10.18420/se2023-ws-15

[4] D. Bilic, D. Sundmark, W. Afzal, P. Wallin, A. Causevic, C. Amlinger, and D. Barkah, "Towards a model-driven product line engineering process: An industrial case study," in *ISEC*, 2020, pp. 1–11. [Online]. Available: https://doi.org/10.1145/3385032.3385043

[5] K. X. Campo, T. Teper, C. E. Eaton, A. M. Shipman, G. Bhatia, and B. Mesmer, "Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature," *Systems Engineering*, 2022. [Online]. Available: https://doi.org/10.1002/sys.21644

[6] J.-C. Chaudemar and P. de Saqui-Sannes, "MBSE and MDAO for early validation of design decisions: a bibliography survey," in *SysCon*. IEEE, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/SysCon48628.2021.9447140

[7] P. De Saqui-Sannes, R. A. Vingerhoeds, C. Garion, and X. Thirioux, "A taxonomy of MBSE approaches by languages, tools and methods," *IEEE Access*, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3222387

[8] R. D. D'Amico, J. A. Erkoyuncu, S. Addepalli, and S. Penver, "Cognitive digital twin: An approach to improve the maintenance management," *CIRP Journal of Manufacturing Science and Technology*, vol. 38, pp. 613–630, 2022. [Online]. Available: https://doi.org/10.1016/j.cirpj.2022.06.004

[9] H. N. Imran and A. Shazia, "The impact of stakeholder communication on project outcome," *African Journal of Business Management*, vol. 5, no. 14, pp. 5824–5832, 2011.

[10] E. Kamburjan, V. N. Klungre, R. Schlatte, S. L. T. Tarifa, D. Cameron, and E. B. Johnsen, "Digital twin reconfiguration using asset models," in *ISoLA*, T. Margaria and B. Steffen, Eds. Springer, 2022, pp. 71–88. [Online]. Available: https://doi.org/10.1007/978-3-031-19762-8_6

[11] T. Le Sergent, F.-X. Dormoy, and A. Le Guennec, "Benefits of model based system engineering for avionics systems," in *ERTS*, 2016. [Online]. Available: https://hal.science/hal-01291938

[12] J. Lu, X. Zheng, A. Gharaei, K. Kalaboukas, and D. Kiritsis, "Cognitive twins for supporting decision-makings of internet of things systems," in *AMP*, L. Wang, V. D. Majstorovic, D. Mourtzis, E. Carpanzano, G. Moroni, and L. M. Galantucci, Eds. Springer, 2020, pp. 105–115. [Online]. Available: https://doi.org/10.1007/978-3-030-46212-3_7

[13] A. M. Madni and M. Sievers, "Model-based systems engineering: Motivation, current status, and research opportunities," *Systems Engineering*, vol. 21, no. 3, pp. 172–190, 2018. [Online]. Available: https://doi.org/10.1002/sys.21438

[14] B. Settles, *Active Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012. [Online]. Available: https://doi.org/10.1007/978-3-031-01560-1

[15] V. Singh and K. E. Willcox, "Engineering design with digital thread," *AIAA Journal*, vol. 56, no. 11, pp. 4515–4528, 2018. [Online]. Available: https://doi.org/10.2514/1.J057255

[16] J. Suryadevara and S. Tiwari, "Adopting MBSE in construction equipment industry: An experience report," in *APSEC*. IEEE, 2018, pp. 512–521. [Online]. Available: https://doi.org/10.1109/APSEC.2018.00066

[17] H. Vangheluwe and G. Vansteenkiste, "Development of an automatic object-oriented continuous simulation environment," *International Journal of General Systems*, vol. 19, no. 3, pp. 263 – 278, 1991.

[18] D. D. Walden *et al.*, *Systems engineering handbook: A guide for system life cycle processes and activities*, 4th ed. Wiley, 2015. [Online]. Available: https://doi.org/10.1002/j.2334-5837.2015.00089.x