Design Science

A quantitative model of hierarchical product design

Ferdinand Wöhr [1,2], Simon Königs[1], Max Stanglmeier[1] and Markus Zimmermann [2]

[1]*BMW Group, Department of Total Vehicle Development, Munich, Germany*
[2]*Technical University of Munich, TUM School of Engineering and Design, Department of Mechanical Engineering, Laboratory for Product Development and Lightweight Design, Garching, Germany*

Abstract

Analysing hierarchical design processes is difficult due to the technical and organizational dependencies spanning over multiple levels. The V-Model of Systems Engineering considers multiple levels. It is, however, not quantitative. We propose a model for simulating hierarchical product design processes based on the V-Model. It includes, first, a *product model* which structures physical product properties in a hierarchical dependency graph; second, an *organizational model* which formalizes the assignment of stakeholder responsibility; third, a *process model* which describes the top-down and bottom-up flow of design information; fourth, an *actor model* which simulates the combination of product, organization and process by using computational agents. The quantitative model is applied to a simple design problem with three stakeholders and three separate areas of responsibility. The results show the following phenomena observed in real-world product design: design iterations occur naturally as a consequence of the designers' individual behaviour; inconsistencies in designs emerge and are resolved. The simple design problem is used to compare point-based and interval-based requirement decomposition quantitatively. It is shown that development time can be reduced significantly by using interval-based requirements if requirements are always broken down immediately.

Keywords: Distributed design, Process simulation, Agent-based modelling

Received 15 May 2022
Revised 31 August 2024
Accepted 03 September 2024

Corresponding author
Ferdinand Wöhr
ferdinand.woehr@icloud.com

Des. Sci., vol. 11, e4
journals.cambridge.org/dsj
DOI: 10.1017/dsj.2024.37

Introduction

The product development process of a company is critical for its business success and ability to compete (Ulrich & Eppinger 2016). Reducing development time and cost while increasing product quality enhances the economic efficiency of an organization. Unfortunately, it is difficult to quantitatively predict the influence of organizational or procedural changes on the development time or product quality in large companies. In vehicle development, for example, thousands of designers simultaneously work on new products. This makes it very challenging to predict the effects of new team structures or task sequences. Knowing the consequences of such procedural or organisational changes in advance, however, is essential for the mitigation of economic risk.

Predicting design process performance is difficult because of the complexity of product development. Products and design processes are complex due to many design variables that are present, many design goals that need to be satisfied, and

many stakeholders that are involved (Summers & Shah 2010; Lindemann *et al.* 2009). In addition to that, the hierarchical structure of products and organizations present in *traditional* product design (Panchal 2009) can affect the complexity as well. This can make the analysis of product design even more difficult. A process model that puts special attention on the hierarchical decomposition of products and the flow of design information is the V(ee)-Model (of *Systems Engineering*) (SE) (Haskins 2006; VDI 2206 2004; Blanchard 2004). As a procedural process model, however, it does not allow any quantitative analysis. In general, it can be observed that quantitative approaches for the analysis of design processes under explicit consideration of product-related and organizational hierarchy are rare. A holistic approach in this regard would, for example, allow for testing frameworks that are focused on improving hierarchical product design. One such framework, for example, is *Solution Space Engineering* (SSE) (Zimmermann *et al.* 2017; Rötzer *et al.* 2020, 2022b).

The overall objective of this paper is to develop a quantitative approach for the analysis of design processes under explicit consideration of product-related, and organizational hierarchy. Clarkson and Eckert (2005) confirm that hierarchically decomposed products and processes are a potential area for future research.

Using the actual development process as a test bed is one way to examine and study design processes. This, however, might disturb regular design activities and cause an increase in development time and/or cost. There are two other ways to quantitatively assess the performance and efficiency of product design processes: first, *experimental studies* under laboratory conditions where human subjects are asked to solve parameter design problems (Hirschi & Frey 2002; Grogan & de Weck 2016). This way most boundary conditions can be controlled while human behaviour is represented quite realistically. A limiting factor is often the required sample size leading to a high number of test repetitions and human subjects that must participate. This is also the reason why such studies are usually performed at scientific institutes rather than at companies, even though some companies have adopted the use of laboratory experiments lately to learn more about and optimize their product design processes (Wöhr *et al.* 2023). A second option is the use of *quantitative models* (Smith & Eppinger 1997a; Clarkson & Hamilton 2000; Yassine *et al.* 2003; Levardy & Browning 2009; Maier *et al.* 2015; Rebentisch *et al.* 2018). Model-based approaches allow an analysis of many different factors in a relatively short time without having to rely on human subjects. Furthermore, models provide great flexibility in terms of abstraction and modelling depth. Thus, compared to field studies or laboratory experiments analytical process models are a powerful and convenient tool to study design processes, in particular, regarding hierarchy.

Some quantitative approaches such as *Game Theory* (GT) (Lewis & Mistree 1997; Gurnani & Lewis 2008; Devendorf & Lewis 2011), or, *Agent-Based Modelling* (ABM) (Jin & Levitt 1996; McComb *et al.* 2015; Hulse *et al.* 2018; Soria *et al.* 2017; Lapp *et al.* 2019a) can be used to investigate design processes in great detail as stakeholders are modelled as computational entities that interact at some point in time. Research in this field deals with cognition, human decision-making, and communication patterns within teams, for example. A significant advantage, in particular of ABM, is the ability to define features, such as product-related or organizational hierarchy on a *microscopic* level. This is essential for our objective as we assume that the dependencies between product properties and interactions

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

between stakeholders on different hierarchical levels need to be modelled in detail in order to examine the associated mechanisms and effects properly.

None of the existing agent-based models, however, considers product-related and organizational hierarchy in a *generic* way (this will be shown in the literature review section). Generic, in this context, means that hierarchies can be modelled on an arbitrary number of vertical levels. This is key as the V-Model, according to which we are simulating design processes, is not limited to shallow hierarchies. Products and organisations may be decomposed onto multiple levels. Further, it can be assumed that the ability to model deep hierarchical structures provides an advantage when analysing the actual dynamics of product design processes.

To address this issue, we propose a model to simulate hierarchical design processes. This includes product properties organized in a hierarchically structured dependency graph and computational agents (the stakeholders in an organization) that are responsible for certain areas of that graph. The model allows the examination of different factors, such as the top-down specification of requirements (point-based versus interval-based design targets). It can be seen as a test bed for the quantitative analysis of hierarchical product design processes. Simulation results could foster the understanding of key mechanisms and improve the performance of product design processes in industrial practice.

This study begins with a *Literature review* that explores the current state of the art in ABM. Specific focus is put on models that consider some form of product-related and/or organizational hierarchy. Thereafter, we state the *Research goal* pursued in this paper. Then, the *Methodology* (research approach) of this work is presented. In the following, the *Proposed model* is introduced by providing a detailed explanation of its four core elements: the product model, the organizational model, the process model and the actor model. Next, we apply the model to an *Example problem* to observe and learn about its general behaviour. A study on the influence of point-based and interval-based requirement formulation is performed afterwards. Finally, a *Discussion* reflects the simulation results and the research approach before a *Conclusion* is drawn and an *Outlook* is given.

Literature review

Agent-based modelling

This review examines existing agent-based models that consider product-related and/or organizational hierarchy. In general, *hierarchy* refers to systems in which "*members of an organization or society are ranked according to relative status or authority*" (Simpson & Weiner 1989). With respect to products, hierarchical structure is usually considered as the arrangement of parts, modules, functions, or product properties according to their dependency. With respect to organizations, hierarchical structure is usually considered as the arrangement of people, teams, or departments according to their managerial authority. It can be assumed that if organizations are set up hierarchically, the information flows adapt accordingly. This implies that the overall transfer of design knowledge is separated into local sequences of information flow between the stakeholders on the various levels.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

In order to assess the existing agent-based models we categorize their ability to model product-related and organizational hierarchy as follows: if the product- or organizational entities are modelled on three or less vertical levels it is considered a *shallow* hierarchy. If product- or organizational entities are modelled on more than three vertical levels it is considered a *deep* hierarchy. Deep hierarchies imply that some kind of generic formalism (to handle vertical interactions) is used.

Agent-based models used in engineering design can generally be divided into two groups: some models attempt to mimic product design processes as realistically as possible. Their goal is to study the mechanisms and dynamics of actual product design processes in which humans perform most of the development work. A list of selected models of this type is given in Table 1. Other agent-based models are (rather) focused on design support, design optimization and process automation. Popular examples of this type are A-Design (Campbell *et al.* 1999, 2003; Cagan *et al.* 2005), the Palo Alto Collaborative Testbed (PACT) (Cutkosky *et al.* 1993), the Shared Dependency Engineering project (SHADE) (McGuire *et al.* 1993), SHADE (Toye *et al.* 1994), and, the Team Knowledge-based Structuring model (TEAKS) (Martinez-Miranda *et al.* 2006, 2012). In most of those models' agent behaviour, decision-making and general problem-solving are modelled in such a way that designers are supported in the best way possible or that design processes are optimized to the maximum. This kind of modelling does not necessarily reflect how design processes actually work. As a consequence, we exclude these agent-based models from our further analysis even though some of them consider deep hierarchical structures (Ambrosio *et al.* 1996; Danesh and Jin *et al.* 2001; Liu *et al.* 2004; Nahm & Ishikawa 2004; Wang *et al.* 2012).

An established and well-known agent-based model for the analysis of design processes is the so-called *Virtual Design Team* (VDT) (Jin & Levitt 1996; Kunz *et al.* 1998; Levitt *et al.* 1999, 2012). The VDT models design work to be done as a precedence network of design tasks where each task requires a certain amount of time (that is effort) from an agent to be completed. The agents that are part of a hierarchical organization in which they send and receive messages along formal lines of authority are assigned to specific sets of tasks. Hence, the VDT includes a rich representation of organizational hierarchy whereas we cannot identify any product-related hierarchy. The tasks do not include technical dependencies.

The approach from Olsen *et al.* (2009) is similar to the VDT. However, in this case, agents are also responsible for domain-specific design variables that are intended to represent distinct product properties on which agents need to find a trade-off if shared among multiple agents. Since those variables affect system-level variables there appears to be a shallow product-related hierarchy. In comparison to the deep hierarchical network of agents in the VDT, however, this model seems to include only two roles (vertical levels): system designers (top) and subsystem designers (bottom). Hence, we suspect a shallow organisational structure.

An agent-based model that focuses on human cognition and individual search behaviour is the *Cognitively-Inspired Simulated Annealing Teams* model (CISAT) (McComb *et al.* 2015, 2017; Raina *et al.* 2019). This approach considers agents that are solving design tasks by using simulated annealing. The meta-parameters of this algorithm correlate with human search behaviour. The design tasks solved by the agents, who share their design solutions within a team without a hierarchical structure, reflect realistic design tasks. In McComb *et al.* (2015), for example, the agents design a truss by manipulating a set of beams and joints so that a specific

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Table 1. Agent-based models used to simulate (hierarchical) product design processes

Framework/theme	Reference	Hierarchical modelling of	
		Product	Organisation
The Virtual Design Team (VDT)	Jin and Levitt (1996); Kunz *et al.* (1998); Levitt *et al.* (1999, 2012)	—	(D)
NASA's JPL/Team X	Olsen *et al.* (2009)	(S)	(S)
Cognitively Inspired Simulated Annealing Teams (CISAT)	McComb *et al.* (2015, 2017); Raina *et al.* (2019)	(S)	—
Complex System Integrated Utilities Model (CAESIUM)	Meluso *et al.* (2019)	(S)	—
Principal-Agent Theory	Vermillion *et al.* (2020); Safarkhani *et al.* (2020)	(S)	(S)
Negotiation/Argumentation Conflict Resolution	Sycara *et al.* (1991); Jin *et al.* (2004)	(D)	—
Single Function Agents (SiFA)	Berker and Brown (1996)	(S)	(S)
Game Theory (GT)	Lewis and Mistree (1997); Gurnani and Lewis (2008); Devendorf and Lewis (2011)	(S)	—
Knowledge Perspective	Zhang and Thomson (2019)	(D)	(S)
Coevolutionary Algorithm	Soria *et al.* (2017)	(S)	—
Communication/Social Learning	Singh *et al.* (2013)	—	(S)
Reinforcement Learning of Agents	Hulse *et al.* (2018)	(S)	—
Mass-Collaborative Product Development (MCPD)	Panchal (2009); Le and Panchal (2011)	(D)	—
Teamwork/Team Dynamics	Singh *et al.* (2019, 2021, 2022)	(S)	(S)
Kirton Adaption-Innovation-Inventory agent-based organizational optimization model (KABOOM)	Lapp *et al.* (2019a, 2019b)	(D)	—
Integrated Product Teams (IPT)	Crowder *et al.* (2008, 2009, 2012)	—	(S)
Management Science/NK models	Loch *et al.* (2003); Rivkin and Siggelkow (2003); Mihm *et al.* (2003, 2010)	(S)	(D)
Agent Model for Planning and Research of Early Design (AMPERE)	Fernandes *et al.* (2017)	—	(D)
Actor-based Signposting (ABS)	Hassannezhad *et al.* (2015, 2019)	(D)	—
Design Information Flow Simulation (DiFS)	Christian (1995) Christian and Seering (1995)	—	(S)
New Product Development (NPD)	Cardinal *et al.* (2011)	—	(S)
Prior versions of our model	Wöhr *et al.* (2020a, 2020b)	(S)	(S)

Abbreviations: (S) shallow hierarchy = entities on three or less vertical levels; (D) deep hierarchy = entities on more than three vertical levels.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

strength-to-weight ratio is reached. This indicates that a shallow product-related hierarchy is considered.

Meluso *et al.* (2019) propose a model to study miscommunication in complex system design. Their approach called *Complex System Integrated Utilities Model* (CAESIUM) includes a network of interdependent artifacts (that is design variables) each of which is connected to a separate objective function. Each agent controls an individual artifact and optimizes the associated objective function while sharing estimates of the latest variable value with others. The resulting network of agents has no formal hierarchy whereas the differentiation into variables and objective quantities can be considered as a shallow product-related hierarchy.

The *Principal-Agent Theory* is another approach used to develop agent-based models. In Vermillion *et al.* (2020) and Safarkhani *et al.* (2020), for example, the theory is applied to study the interactions between system designers (principals) and subsystem designers (agents). Both are collaborating hierarchically. System designers usually provide objectives or requirements to subsystem designers who then try to optimize or fulfil them. The design problems solved by the agents are context-free utility functions that depend on some (abstract) variables. The models therefore include shallow product-related and organizational hierarchies.

Negotiation, argumentation and conflict resolution are the main themes of the models presented by Sycara *et al.* (1991) and Jin *et al.* (2004). Agents in this case solve design problems resulting from technical interdependencies by suggesting, evaluating, accepting or rejecting design proposals to/from other agents on the same hierarchical level. In order to rate design proposals or resolve discrepancies agents possess quantitative belief structures (dependency graphs including design parameters) (Sycara *et al.* 1991) and engage in multi-level negotiation processes with design issues on various levels influencing each other (Jin *et al.* 2004). This can be seen as a deep hierarchical product structure.

In Berker and Brown (1996) the concept of *Single Function Agents* (SiFAs) is established. SiFAs have specific functions, designated design targets and distinct perspectives from which they operate when debating about design parameters with other agents. While some select or estimate parameter values others criticize or praise them from their individual point of view. Thus, the model involves some level of managerial authority (as some agents only propose values and some only criticize values) and some level of product-related hierarchy as design targets and points of view could also be called design variables and objective quantities.

Other models utilize *Game Theory* (GT) which is quite similar to ABM. Players in GT typically represent individual and autonomous decision-making entities (much like typical agents). In Lewis and Mistree (1997), Gurnani and Lewis (2008) and Devendorf and Lewis (2011), for example, players control separate subsystems by manipulating the corresponding design variables. Simple equations are used for describing the dependency between subsystem- and system level. Depending on the given process architecture (sequential or parallel) players communicate information about their latest designs to other players on the same hierarchical level. Hence, we assume no organizational hierarchy while the design problems indicate a shallow hierarchical structure.

Zhang and Thomson (2019) suggest a model with a particular focus on design knowledge. Their approach considers a multi-level network of product functions where each function requires certain knowledge from designers to be completed.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Besides working on functions designers may consult experts or managers to solve design problems and interface issues. This shows that the model includes a deep hierarchical product structure and a shallow organizational hierarchy (designers, experts and managers).

A model in which teams of individual agents solve complex design problems by applying a coevolutionary algorithm can be found in Soria *et al.* (2017). Here, agents responsible for different subsystems optimize local utility functions based on specific design variables. Complete vehicle designs are created by assembling the individual design solutions from the agents without a central design authority managing design conflicts. Thus, it can be assumed that the model only considers a shallow product-related hierarchy.

The agent-based model introduced by Singh *et al.* (2013) is used to study the effect of communication within design teams. In this case, projects are considered as a hierarchical network of design tasks each of which requires a certain amount of time (effort of agents) to be completed (similar to the VDT). Communication within design teams, which consist of agents and a team leader, depends on a team's structure (flat, distributed, functional). Hence, the organisation considered by the model is characterized by a shallow hierarchy whereas a product-related hierarchy cannot be observed since the tasks are abstract units of work.

Design agents in the model proposed by Hulse *et al.* (2018) are equipped with a reinforcement learning algorithm. In order to solve a complex real-world design problem composed of a variety of design variables and objective quantities agents possess distributed authority (that is control) over design variables. Therefore, there appears to be no formal hierarchy between the agents whereas the product model (the design variables and the objective quantities) involves a shallow hierarchical structure.

While most agent-based models used in engineering design focus on traditional (top-down based) product development Panchal (2009) and Le and Panchal (2011) suggest an approach for simulating *Mass-Collaborative Product Development* (MCPD). In this case, products are modelled as *modules* that are organized in a directed (hierarchically structured) graph. The completion percentage of each module is increased if agents spend time working on it. The agents decide themselves (without authority or hierarchy) on what modules to work (next). This is typical for the self-determination of participants in MCPD.

Singh *et al.* (2019, 2021, 2022) introduce a model for the analysis of design teams. In this model, design teams are represented by agents who are searching for the best solution of an analytical two-dimensional function. After each session in which the agents optimize their individual solutions by varying the different design variables of the objective function, some propose their results to the team. If the proposed solution is accepted by the group it is given to a controller agent who evaluates its general acceptability and decides on further steps, e.g., if further design sessions are required. Thus, the model includes a shallow organizational structure and a shallow product-related structure.

A model similar to the CISAT framework is proposed by Lapp *et al.* (2019a, 2019b). The so-called *Kirton Adaption-Innovation Inventory agent-based organizational optimization model* (KABOOM) suggested by the authors can be used to examine design teams in which team members possess different cognitive behaviours. The agents in this model use a simulated annealing algorithm to optimize a given cost function by varying design variables they are assigned to. In each

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

iteration, agents may engage in a pairwise interaction with others and share their current position in the design space. This approach does not consider any organizational hierarchy whereas the product model includes a shallow hierarchical structure.

The agent-based model developed by Crowder *et al.* (2008, 2009, 2012) helps studying *Integrated Product Teams* (IPT). For this purpose, the approach includes designer agents who are working on abstract design tasks and manager agents who are assigning tasks to designer agents. The individual tasks belong to larger units of work. They do not relate to product features such as components, functions or physical product properties. Therefore, we assume that product-related hierarchy is neglected. The consideration of agents with the authority to assign design tasks (managers) indicates a shallow hierarchical structure.

Using agent-based models to study product design processes is also common practice in Management Science. Approaches in this field often include so-called NK models ("*rugged landscapes*"). Those are formal search problems for which agents are asked to find the best possible solution (Loch *et al.* 2003; Mihm *et al.* 2003). NK models usually include local decision variables and local performance functions. Those are assigned to individual agents. Global performance functions are used to integrate individual solutions. In some cases, the shallow product-related hierarchies are coupled with shallow (and deep) organizational hierarchies to examine the dynamics of multi-level search and decision processes (Rivkin & Siggelkow 2003; Mihm *et al.* 2010).

Fernandes *et al.* (2017) present an approach called the *Agent Model for Planning and Research of Early Design* (AMPERE). In this particular model agents with different roles (customer, lead agent, senior designer, junior designer) and design authority develop products by collaborating hierarchically. While top-level development work consists of requesting design solutions, evaluating risks, and directing resources, lower-level development work is modelled as time spent on technical design tasks. For this reason, we assume that this model includes a rich representation of organizational hierarchy while the design tasks processed by the different agents include dependencies between product artifacts only in an indirect manner.

Actor-based Signposting (ABS) developed by Hassannezhad *et al.* (2015) and Hassannezhad *et al.* (2019) is an extension of the original Signposting framework introduced by Clarkson and Hamilton (2000). This model contains agents that are working on parameter-driven design tasks. These tasks consider the dependencies between design variables via so-called *confidence levels.* The agents processing the tasks do not seem to be subject to a formal hierarchy. Thus, it can be assumed that ABS only involves a deep hierarchical decomposition of the product.

The *Design Information Flow Simulation* (DiFS) is an approach suggested by Christian (1995) and Christian and Seering (1995). In DiFS agents with different roles (responsible and assists) work on design projects that are broken down into subprojects. Subprojects are high-level design tasks (*Design*, *Review*, etc.) that do not consider any detailed product information. Responsible and assisting agents handle subprojects to increase their completion while also engaging in review meetings where discussions are held. Thus, we suspect that DiFS only considers a shallow organisational hierarchy.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

An agent-based model for the analysis of *New Product Development* (NPD) is described by Cardinal *et al.* (2011). As in other approaches, development projects in this case consist of abstract design tasks that require a certain amount of effort from agents to be completed. Agents are either project managers, or, employees. Their role defines which tasks they are supposed to work on. In general, it can be assumed that this model only incorporates a shallow organizational hierarchy.

Prior versions of the model presented in this paper can be found in Wöhr *et al.* (2020a) and (2020b). Those earlier modelling attempts already include design variables and agents on a few vertical levels in order to study hierarchical design processes (in a rudimentary way). In Wöhr *et al.* (2020b), for example, we simulate a hierarchical design process consisting of system-level and component-level designers. Both need to collaborate to solve a multi-level design task. As in other studies the earlier versions of our model do not involve a deep hierarchical structure of the product or the organisation.

Summary and conclusion

The previous survey of agent-based models provides three insights: first, many of the existing models consider some (shallow) level of hierarchy with respect to the organisation or product that is studied. Most of the time those shallow hierarchies are associated with simplified design problems or basic organisational structures on two vertical levels required to study some specific research questions. Second, some models consider deep product-related, or, deep organisational hierarchies to account for design problems or managerial structures that spread over multiple levels of dependency or authority. And third, there seems to be no approach that involves a deep hierarchy of both, the product and the organisation. This indicates that there also exists no concept of how to handle hierarchies (product-related and organizational) in a generic way.

Note that other modelling approaches such as Petri nets (Peterson 1981; Belhe & Kusiak 1993) are already set up to account for hierarchical processes (van der Aalst & van Hee 2004). The explicit consideration of product-related, and, organizational hierarchy, however, is not possible yet.

Some methods from multidisciplinary design optimization, such as *Analytical Target Cascading* (ATC) (Kim *et al.* 2003a, 2023b), for example, already possess the capability to model hierarchical product structures and organizations. ATC is also used in industrial design practice, as described by Papalambros and Wilde (2017) and Kang *et al.* (2014). While ABM uses software agents to solve hierarchical design problems ATC decomposes design problems into hierarchical subproblems that are solved individually and coordinated in a specific way.

Research goal

The goal of this paper is to develop and apply a quantitative model for simulating product design processes with deep hierarchical structures. This means it should be possible to model the physical properties of a product and stakeholders within an organization on an arbitrary number of vertical levels.

Based on the developed model the objective is to evaluate the influence of SSE on the performance of design processes quantitatively. This has, to the best of our

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

knowledge, not been done before and could help companies decide whether to implement SSE.

Methodology

Research approach

The model that we present in this paper is developed according to the V-Model of Systems Engineering (Blanchard 2004; Haskins 2006). The V-Model is also part of the design methodology for mechatronic systems (VDI 2206 2004). Thus, our modelling approach is mainly theory-driven. Many assumptions are confirmed by Fernandes *et al.* (2017). They describe an industrial case study in which top-down and bottom-up design according to the V-Model also occurs in actual practice.

Guidance for the transfer of information from the V-Model into a quantitative simulation model is taken from Zimmermann *et al.* (2017) and Maier *et al.* (2022). This is supported by the evidence found in Vermillion *et al.* (2020) and Safarkhani *et al.* (2020). The product model builds on the fundamentals developed by Rötzer *et al.* (2022a).

For exploratory purposes, our model is applied to an example problem where we simulate four different design scenarios (variation of requirement formulation and agent behaviour). We track the required and realized designs identified by the agents while the product structure and the overall design goals are held constant.

Model classification

The (sequential) product development process described by Ulrich and Eppinger (2016), see Figure 1, allows us to illustrate the temporal scope of our model. In this generic process, our approach addresses *System-Level Design*, *Detail Design*, and *Testing and Refinement.* Another reference model can be found in Albers and Mebolt (2007) and Albers *et al.* (2016). According to the framework they suggest called the *integrated Product engineering Model* (iPeM) our model is also applicable to products that are developed over generations. Finally, we want to integrate our model into the frame of reference provided by Pahl *et al.* (2007). With respect to their concept, our model corresponds to *original design*, that is, the realisation of problems by incorporating new solution principles.

Proposed model

Model overview

The quantitative model we propose in this paper consists of four key components: first, a *product model*, which establishes a relationship between physical product

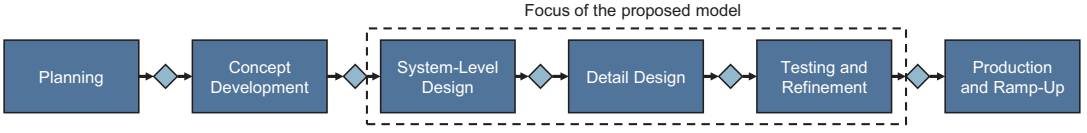


Figure 1. The product development process according to Ulrich and Eppinger (2016).

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

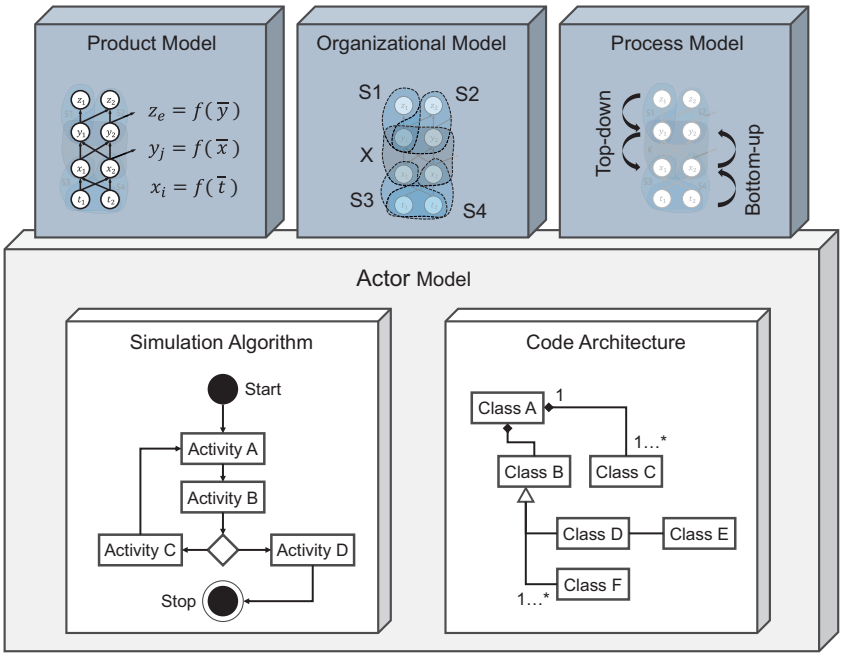


Figure 2. Components of the proposed model.

properties on various hierarchical levels. Second, an *organizational model* which formalizes the assignment of stakeholder responsibility within the product model. Third, a *process model* which describes the top-down and bottom-up information flow between stakeholders. And, fourth, an *actor model* which utilizes an agent-based approach to simulate the combination of product, organization and process. The actor model itself includes two elements: first, a *Simulation Algorithm* which schedules the agents during each iteration and which defines the individual design behaviour of agents. Second, a *Code Architecture* provides the general blueprint for the implementation of the model.

Figure 2 shows an overview of the model presented in this paper, including its four key components. While the product model, the organizational model, and the process model may be seen as the fundamental building blocks, the actor model may be seen as the processing unit.

When simulating a use case the product model, the organizational model, the initial conditions (top-level requirements), and the boundary conditions (limits of the design space) need to be given as inputs. This allows us to study different design problems, and, different responsibility distributions. Further, our model allows us to choose from two different algorithms for agent behaviour. The process model and the remainder of the actor model are fixed.

Product model

Directed graphs consisting of vertices (nodes) and edges (arrows) are commonly used to capture the causal relationships between different elements of a system. In social sciences, for instance, *Directed Acyclic Graphs* (DAG) are used to illustrate

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

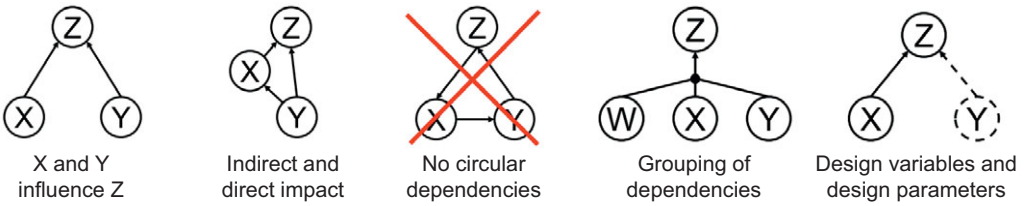


Figure 3. Rules for modelling ADGs (taken with permission from Rötzer et al. (2022a)).

the dependencies between parameters of field experiments (Elwert 2013). In our case, vertices represent the physical properties of a technical system, like the stiffness of a beam, the displacement volume of an engine, or the acceleration of a vehicle. All of them can either be measured or determined by analysis and, thus, assume a numeric value or the value true or false. Technical dependencies between those product properties, derived through analytical formulae or by approximation, are depicted as edges. Their direction illustrates what property influences what other property. In a particular sense, the general idea is that whatever can be measured first will determine what is generated by combining the associated elements. For example, the mass of a single object and the stiffness of a spring can be measured before they are combined into a harmonic oscillator with an eigenfrequency. The mass and the stiffness therefore determine the eigenfrequency, and the eigenfrequency depends on the mass and the stiffness. This type of hierarchical product model is called an *Attribute Dependency Graph* (ADG) (Rötzer *et al.* 2022a). By definition, ADGs do not include circular dependencies also known as *feedback loops*. This prevents scenarios in which cause and effect cannot be distinguished. Basic rules for modelling ADGs according to Rötzer *et al.* (2022a) are depicted in Figure 3. A similar approach for capturing dependencies among design variables and design constraints without including feedback loops is presented by Kusiak (1999). The *Functional Dependence Table* (FDT) described by Wagner (1993), Papalambros and Wilde (2017), Krishnamachari and Papalambros (1997a, 1997b), and Allison *et al.* (2009) is another method for modelling dependencies between product properties. In comparison to ADGs, FDTs do not provide information about directionality.

Identifying design variables and cascading dependencies can be difficult in actual design practice. Hence, the following paragraphs provide a gradual introduction to elementary (real-world) design settings modelled with ADGs:

One-to-one relationship

The simplest ADG possible represents a one-to-one relationship between one design variable and one quantity of interest. Assume, for example, that the displacement volume of an engine (x) has an effect on the power of the engine (y). This system is illustrated in Figure 4(a). A change in the displacement volume of the engine causes a higher or lower power of the engine because it can be measured first, i.e., the displacement volume of the engine can be determined independently whereas the power of the engine is determined based on the engine's displacement volume.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

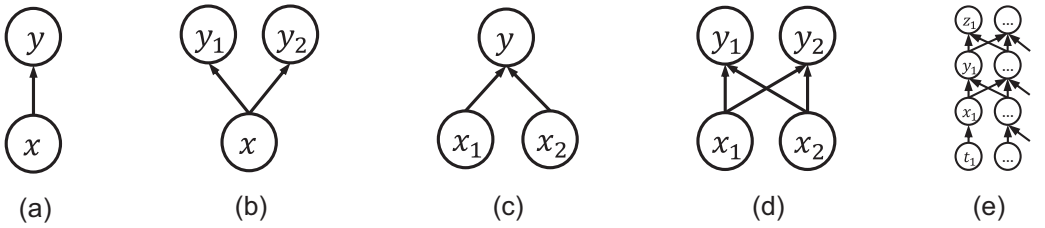


Figure 4. Elementary design settings modelled with ADGs.

One-to-two relationship

Building on the example from before, now assume that the displacement volume of the engine (x) has an influence on the power of the engine (y_1) and on the mass of the engine (y_2). For this case, the ADG is shown in Figure 4(b). This scenario represents a conflict of goals as multiple quantities of interest depend on a smaller number of design variables.

Two-to-one relationship

Two design variables can also influence the same quantity of interest. Assume the total mass of a vehicle (y) depends on the mass of the engine (x_1) and the mass of all other subsystems (x_2), which includes the chassis, the gearbox and the body, for example. Hence, multiple degrees of freedom exist. This system representing a two-to-one relationship is shown in Figure 4(c).

Two-to-two relationship

Assume a technical system where not only the displacement volume of the engine (x_1) but also some material property of the shaft (x_2), for example, has an effect on the power of the engine (y_1) and the mass of the engine (y_2). This two-to-two correspondence is depicted in Figure 4(d). Design problems represented by fully coupled ADGs like this are often difficult to solve for single designers and design teams (Hirschi & Frey 2002; Grogan & de Weck 2016).

L-to-M-to-N relationship

Designing complex systems usually involves a wide variety of product properties that are located on multiple hierarchical levels. ADGs are capable of representing such systems by incorporating a sufficient level of detail. In order to demonstrate that assume that the power of the engine (y_1) also influences the acceleration of the vehicle (z_1) and the height of the cylinders (t_1) has an impact on the displacement volume of the engine (x_1). This ADG is illustrated in Figure 4(e) where further product properties on each hierarchical level are shown as well. In general, there is no limit on how many properties another property can influence, or how large an ADG can be. This only depends on the scope of the investigation.

Bottom-up mappings

Identifying the current state of a specific product property based on the state of all influencing product properties below is called *bottom-up mapping* (Zimmermann *et al.* 2017). This can be done by using physical models, mathematical surrogate models, hardware experiments, or expert assessments, for example.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

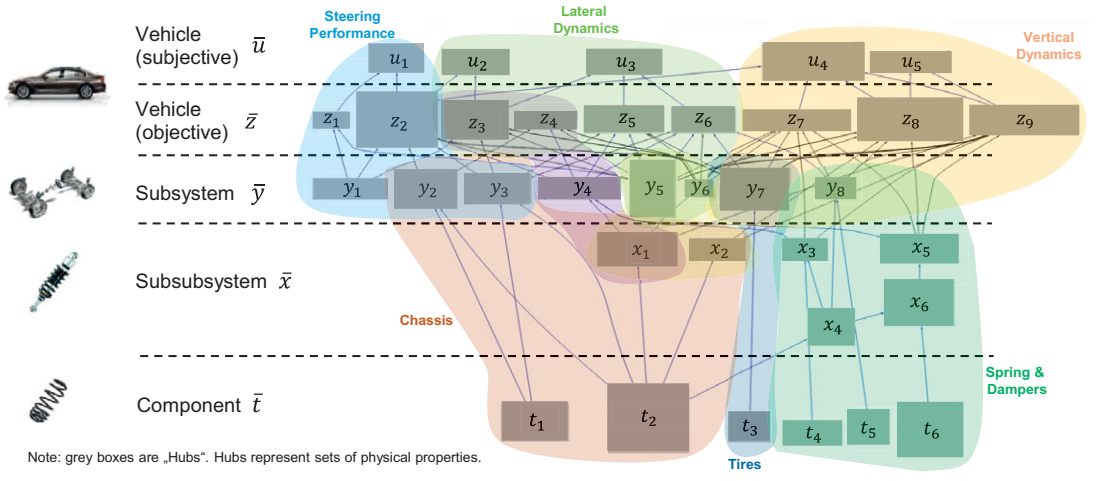


Figure 5. An ADG from an automotive development project representing a design scenario that could be simulated with our model. Reproduced (with permission) from Rötzer et al. (2022a).

Examples of *application*

Using ADGs to model products can result in complex dependency graphs that illustrate the technical design challenge at hand. Figure 5, for example, shows an ADG from an automotive development project (Rötzer *et al.* 2022a). In this case, subjective vehicle properties (e.g. cornering behaviour) depend on objective vehicle properties (e.g. yaw dynamics) which, in turn, depend on subsystem-level properties (e.g. toe angles), and so on. The coloured areas indicate the association between properties and disciplines or components. Note that this ADG represents the vehicle design at a specific moment in time. Conceptual changes would mean that properties are being added or removed. This transient aspect of design is not considered by the model we propose.

Others have applied ADGs to control systems (Zare *et al.* 2017; Korus *et al.* 2018), crash structures (Zimmermann *et al.* 2017), engine mounts (Zimmermann *et al.* 2017; Wöhr *et al.* 2020a) and robots (Krischer & Zimmermann 2021).

Organizational model

The coloured areas shown in Figure 5 also represent areas of responsibility. Those reveal which stakeholder is responsible for which set of product properties. Being *responsible*, in this context, means that a stakeholder manages and controls those properties to drive a product towards its desired design objectives. Figure 6 shows a generic (formalized) ADG with five different areas of responsibility that we will use to explain the further model (Maier *et al.* 2022),

Allocation of responsibility

The product properties at the bottom of an area of responsibility are called *design variables* (x_1 and x_2 for X), and the ones at the top are called *quantities of interest* (y_1 and y_2 for X). A design variable may be a quantity of interest in another area of responsibility below. Due to simplicity, we do not consider intermediate product properties.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

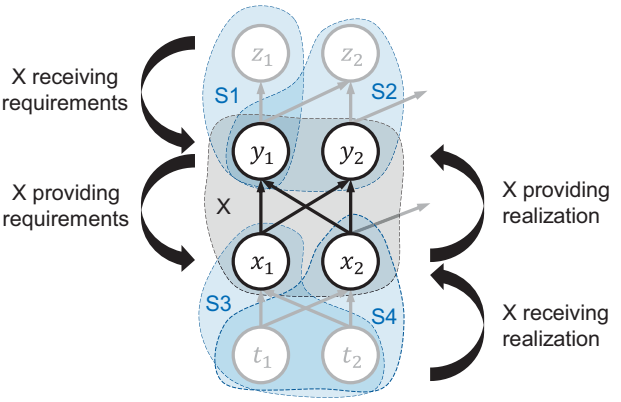


Figure 6. A (generic) ADG with top-down decomposition of requirements and bottom-up feedback of realizations (Maier et al. 2022).

A product property must always be a design variable in an area of responsibility where it influences another property. Observe y_1, for example, which has to be a design variable within the area of responsibility that belongs to S2 as it influences z_2. This might lead to situations where product properties are design variables in multiple (overlapping) areas of responsibility. Each of those properties, however, can only be a quantity of interest in exactly one area of responsibility. This makes sure that the fulfilment of all requirements assigned to a certain quantity of interest is handled by a single stakeholder.

It is assumed that each stakeholder is only informed about the properties and dependencies in their own area of responsibility. The entire structure of an ADG and the distribution of responsibility is unknown to the individual stakeholders. Each stakeholder is only aware of in how many areas of responsibility above the own quantities of interest are a design variable. This information is required as it defines the number of stakeholders for each quantity of interest a stakeholder has to exchange design information with.

Process model

We assume that the socio-technical system shown in Figure 6 exhibits an iterative top-down and bottom-up flow of design information as described by Maier *et al.* (2022). For this process, we will present a quantitative model in the following.

Types of design information

Two primary types of design information can be linked to every product property of an ADG: *requirements* and *realizations.* They are used to describe *desired* and *realized* states of properties that are located at the organizational interfaces (the overlapping areas) between multiple areas of responsibility. Our definition is:

- A **requirement (REQ)** describes the desired state of a product property. It can be evaluated as true (satisfied) or as false (not satisfied). Even though, there are many different kinds of requirements, like functional, interface or constraint (Kapurch 2007; Maier *et al.* 2022) we assume that all of them can be described

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

quantitatively based on a lower and upper limit ($x_{i,lb}, x_{i,ub}$). Requirements can be formulated as interval-based ($x_{i,lb} < x_{i,ub}$) or point-based ($x_{i,lb} = x_{i,ub}$). One-sided requirements ($x_{i,lb} = \infty \vee x_{i,ub} = \infty$) are considered a special case of interval-based. In particular interval-based requirements as used in SSE are gaining attention lately (Zimmermann *et al.* 2017) as they allow more design information to be shared between stakeholders and, therefore, increase design freedom. Note that SSE is similar to Set-Based Concurrent Engineering (SBCE) (Sobek *et al.* 1999). In SBCE, also called Set-Based Design (SBD) (Toche *et al.* 2020), however, alternative designs are continuously removed until the final design is selected whereas in SSE sets of good designs are maximized (Zimmermann & von Hoessle 2013). A comparison between SSE and SBD can be found in Fender *et al.* (2014).

- A **realization (REAL)** describes a *realized* state of a product property that can be described quantitatively as well. Every component design ($\mathbf{t}_r$) is a set of realizations ($t_{1,r}, t_{2,r}$) linked to the associated product properties on the bottom level. Realizations of product properties above can be quantified through bottom-up mappings, e.g. $x_{i,r} = f(t_{1,r}, t_{2,r})$. In this paper, we consider point-based realizations only.

A requirement is assumed to be satisfied if the associated realization lies between the lower and upper limit. This means the following condition has to be satisfied

$$x_{i,lb} \leq x_{i,r} \leq x_{i,ub}. \tag{1}$$

A set of requirements assigned to some product properties on the same hierarchical level is called a *required design*. A set of realizations linked to some product properties on the same hierarchical level is called a *realized design*.

Flow of design information

While requirements are usually decomposed *top-down* (in Figure 6: from **z** to **t**), realizations normally propagate *bottom-up* (in Figure 6: **t** to **z**). In the following, both processes, top-down and bottom-up, will be explained separately. In design practice, they often happen simultaneously.

Top-down: In the proposed model, a stakeholder formulates requirements on those properties that serve as design variables in his or her area of responsibility. They need to be formulated in such a way that, if satisfied, the requirements on the stakeholder's own quantities of interest are satisfied. Stakeholder X, for example, receives requirements on y_1 and y_2 from stakeholder S1 and S2 as both of them try to reach their design goals, i.e., requirements, which are formulated with respect to z_1 and z_2. Those requirements on y_1 and y_2 may be point- or interval-based. As y_1 is a design variable for S1 and S2, it has two requirements linked to it. Hence, X has to *harmonize* the requirements on y_1 first meaning that based on all lower and upper bounds a *harmonized requirement* has to be determined. In our model, this is always done for properties which are design variables in multiple areas of responsibility. Thereafter, X derives requirements on x_1 and x_2 which, if realized, support the fulfillment of the harmonized requirements on y_1 and y_2. Here, X may choose point- or interval-based requirements for x_1 and x_2. Those requirements on x_1 and x_2 are then forwarded to S3 and S4 to serve as their design goals.

This top-down flow of design information (requirements) is a generic process, which is applicable on all hierarchical levels of an ADG. Thus, S3 and S4 execute the

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

same process as described above. This time, however, the requirements they derive on their own design variables are not passed on to the next lower level but turned into realizations of t_1 and t_2 as both of those properties are located on the bottom level of the ADG. Hence, deriving requirements on properties located on the lowest level of an ADG is the same as selecting a specific design.

Bottom-up: Based on the realizations linked to t_1 and t_2, S3 and S4 determine the realizations of x_1 and x_2 through bottom-up mappings. We refer to this design activity as *quantifying*. In this study, realizations are always defined as point-based. After quantifying and submitting the realizations to X, S3 and S4 assess whether the requirements on their own quantities of interest are satisfied or not based on the realizations of x_1 and x_2. We refer to this design activity as *evaluating*. Note that this sequence of activities is a strong modelling assumption. The stakeholders S3 and S4 could, for example, also evaluate before submitting the results to X.

This bottom-up flow of design information (realizations) is, like the top-down flow of requirements, a generic process that takes place on multiple hierarchical levels of an ADG. X, for example, uses the realizations assigned to x_1 and x_2 to determine the realizations of his or her own quantities of interest. Afterwards, X evaluates, whether the requirements assigned to y_1 and y_2 are satisfied. If they are not fulfilled X reformulates the requirements on his or her own design variables considering the realizations of x_1 and x_2 in order to facilitate the design work of S3 and S4. This *adaptive* requirement redefinition under consideration of bottom-up information (realizations of the own design variables) is also performed by S1, S2, S3 and S4 as they receive feedback for their design variables that, in turn, do not satisfy the requirements assigned to their own quantities of interest.

Summary: Especially in the case of large ADGs, which are typical for industrial development projects, the simultaneous top-down decomposition of requirements and bottom-up feedback of realizations (on multiple hierarchical levels) leads to complex system behaviour. As in many other systems, such as the stock market or the traffic system, this behaviour is usually not controlled by a centralized agent, but rather a result of the individual entities that follow a (most of the time simple) set of rules and interact in a certain way. Hence, our model of hierarchical design processes is well suited to be modelled with an *agent-based* approach.

Actor model

Using simulation models to investigate product design processes has some major advantages over conducting field studies or laboratory tests with human subjects: first, simulated design processes are repeatable, second, many simulations can be performed thanks to less limited resources, third, there is no interference with an ongoing real-world development process that participants of a field study may be involved in. The model we propose is based on computational entities (or *agents*) who need to collaborate in order to identify a good product design. The following chapter provides a technical and detailed description of this model.

Fundamentals of agent-based modeling

Many systems which are composed of a large number of autonomous, interacting entities exhibit a dynamic behaviour that is characterized by self-organization, pattern formation, and emergence (Macal & North 2008). Typical examples are supply chains, financial markets or the spread of epidemics. Modelling them on a

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

global scale based on a set of state variables for the complete system is often not sufficient when mechanisms on a very detailed level want to be studied.

Agent-based models provide a natural way of describing such systems based on a set of computational entities, so-called *agents*, that usually follow a (simple) set of rules. Their interaction can lead to complex patterns (structural phenomena that repeat themselves over space and time) which, in general, are unforeseeable by just analysing the agents' rules alone. Even though there exists no universal definition of what constitutes an *agent-based model* or even an *agent*, it can be assumed that the following elements are usually included (Macal & North 2010): *agents*, i.e., computational entities which have attributes and executable methods. In our case, agents can be individual designers, design teams or departments. Each agent has a role, which is linked to a certain area of responsibility. In that way, the model is generic and allows stakeholder heterogeneity. And *relationships*, i.e., ties among the agents. In our case, agents are tied if the areas of responsibility their roles are assigned to overlap.

Agent-based models in general are suitable if real-world systems which cannot be described conventionally need to be analysed on a computational basis. They are especially useful in the case of complex processes with many interacting entities.

System boundaries and simplifications

As we focus on the development phases of System-Level Design, Detail Design, and Testing and Refinement, we assume that a first technical concept has been derived (selection of subsystems and components) but no final subsystem and component designs have been chosen yet. This has the following three implications: first, we assume the structure of an ADG (which often evolves during early design phases) to be *given* and to be *static.* Static means that there is no time-dependent change in the number of subsystems or components (no new properties or dependencies). Second, we assume that the requirements assigned to the product properties at the top level of an ADG are given. Those are usually defined directly at the beginning of development processes. Third, we assume that the realizations linked to the product properties at the bottom level of an ADG can always be manipulated. In industrial practice, they are usually fixed at some point in time (late in product development) to start the production of the manufacturing systems.

Software design and code architecture

The software implementation of the proposed model is object-oriented. An object is an instance of a class. A class is a generic blueprint that includes all attributes and methods necessary to describe an object and its behaviour (Weilkins 2007).

Figure 7 shows the Unified Modelling Language (UML) class diagram of the quantitative model presented in this paper. It is the extension of an earlier model version presented by Wöhr *et al.* (2020b). Other UML class diagrams which have been used to describe agent-based models in engineering design include similar classes but differ in the way how they describe product information or individual agent behaviour (Fernandes *et al.* 2017). Below all classes are written in italics.

In the proposed model every *Process Simulation* includes *Agents*, a *Scheduler* which coordinates the order in which the agents are activated each time step, and a *Design Process.* The latter has a *Public Database* in which the complete *ADG* of the current product design is stored. This might be thought of as a central repository

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

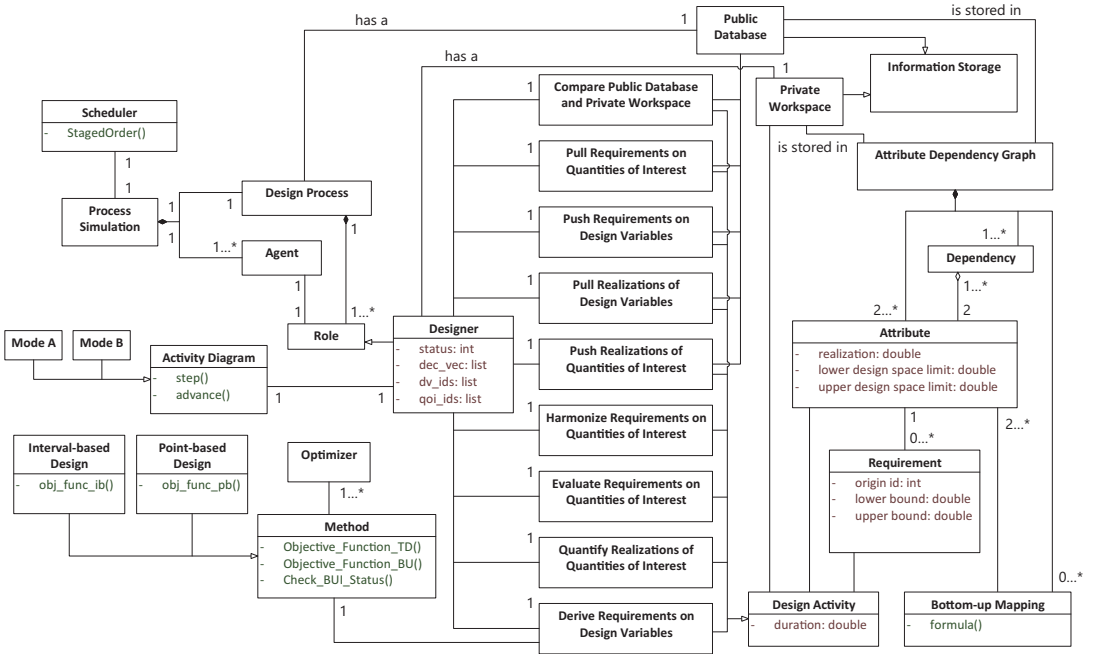


Figure 7. UML (Unified Modelling Language) class diagram (data model) of the quantitative model.

for design information that needs to be documented and shared. All properties of an ADG are *Attributes*. Each of them has a repository to store the realization of that product property, denoted as $x_{i,r}$. The minimum and maximum (permissible) value of each realization is defined by the corresponding design space which is also stored directly within each attribute. The quantitative relationships between properties, formally expressed by $y_i = f(\mathbf{x})$, are *Bottom-up Mappings*. Executing a bottom-up mapping means evaluating the given function. The *Dependencies* of an ADG are not instantiated when running a simulation, as there is no explicit usage. Design targets on product properties are *Requirements*. They include a lower and an upper limit, denoted as $x_{i,lb}$ and $x_{i,ub}$, which are used to express the desired state of that product property. Both are possible, point-based and interval-based requirement formulation.

Every agent that is part of the process simulation, whether it is an individual designer, a design team or a department has the *Role* of a *Designer*. Each designer has a status indicating his or her current occupation (busy or not busy), a decision vector that governs the behaviour, and, an area of responsibility that he or she is assigned to. A copy of all attributes and dependencies located within this area of responsibility is stored in the *Private Workspace* every designer has. This can be thought of as a local *Information Storage* where each designer tests and optimizes the performance of its assigned component or subsystem without interfering with others. The private workspace covers the entire area of responsibility.

During a process simulation, each designer performs *Design Activities*. Those can have three major goals: first, **compare** the private workspace with the public database in order to check for any new design information (requirements and/or

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

realizations) related to the own attributes (*Compare Public Database and Private Workspace*). Second, **transfer** design information from the public database into the private workspace (*Pull Realizations of Design Variables*, *Pull Requirements on Quantities of Interest*), or, the other way, from the private workspace into the public database (*Push Realizations of Quantities of Interest*, *Push Requirements on Design Variables*). Third, **work** on the attributes in the private workspace, which includes harmonizing the requirements that are linked to the quantities of interest (*Harmonize Requirements on Quantities of Interest*), to decompose those harmonized requirements into new requirements on the design variables (*Derive Requirements on Design Variables*), to quantify the realizations of the quantities of interest based on the realizations of the design variables received from below (*Quantify Realizations of Quantities of Interest*), or, to compare the harmonized requirements on the quantities of interest with the achieved realizations (*Evaluate Requirements on Quantities of Interest*). The decomposition of requirements has two different *Methods* for point- and interval-based requirement formulation.

In each iteration of a simulation, a designer decides which design activities to perform based on the design information in the private workspace and the public database as well as its current occupation. The decision-making of a designer is governed by his or her *Activity Diagram*. It is a deterministic model to describe behaviour in a rational way. We provide two *Modes* of the activity diagram to account for two different types of design behaviour.

Output information and process metrics

After each simulation run, the requirements and realizations linked to all product properties in each iteration can be analysed. This allows us to study the evolution of designs on different hierarchical levels in a very detailed way. In addition, typical performance metrics, or *characteristics*, to determine the success of development processes (Ulrich & Eppinger 2016) can be assessed based on the requirements and realizations assigned to the product properties on the top level. This includes, for example, the *product quality*, that is, a metric of how *good* a product currently is. In our case, this is defined as the normalized distance between all requirements on the system level and the achieved realizations. Also, the *development time*, that is, a measure of how *fast* a product is developed. In our case, this is defined as the number of iterations needed to identify good component designs, i.e., realizations assigned to the properties on the bottom level that, if propagated to the top, satisfy all system-level requirements.

Formal description of design information

We describe design information in an area of responsibility with design variables denoted as x_i with index $i \in \{1,\ldots,h\}$ and quantities of interest denoted as y_j with index $j \in \{1,\ldots,k\}$ as follows: for each quantity of interest y_j the requirements are denoted as $\mathbf{y}_{j,lb}$ (all lower bounds) and $\mathbf{y}_{j,ub}$ (all upper bounds), the harmonized requirement is denoted as $y_{j,lb,H}$ (lower bound) and $y_{j,ub,H}$ (upper bound), and, the realization is denoted as $y_{j,r}$. For each design variable x_i the requirement is denoted as $x_{i,lb}$ (lower bound) and $x_{i,ub}$ (upper bound) and the realization is denoted as $x_{i,r}$. Moreover, the public database is denoted as PD, and the private workspace is denoted as PW.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

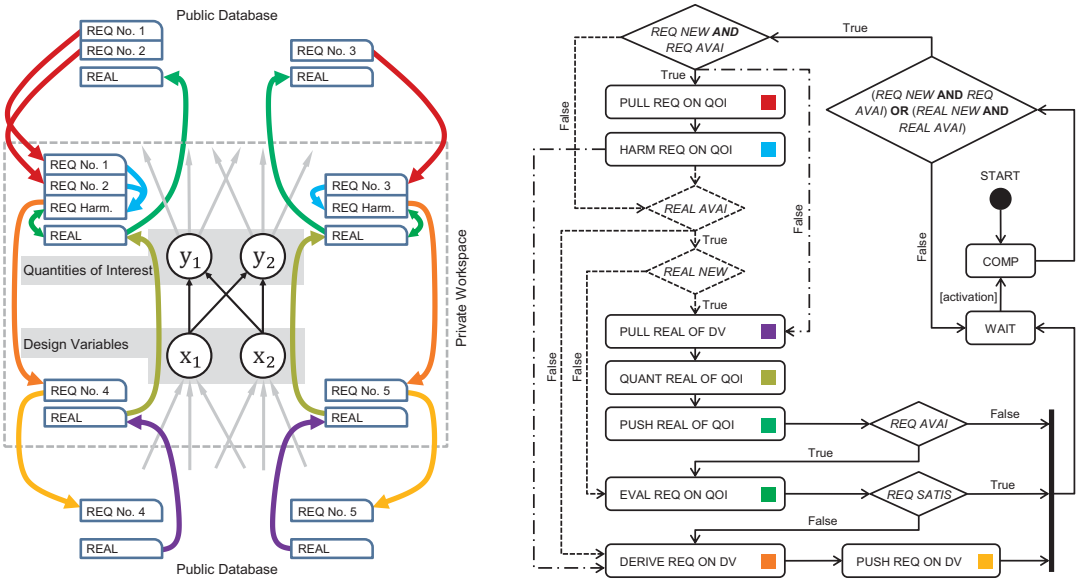


Figure 8. Visualized design activities (left) and activity diagram (right) in Mode A: — ; – – and Mode B: — ; – · –.

Agent scheduling and decision-making

The process simulation consists of two nested algorithms: a time-discrete routine in which all agents in their role as designers are activated once per iteration, and, a UML activity diagram, depicted on the right side of Figure 8, which determines the behaviour of designers when they are activated. The activity diagram connects the decision vector every designer has shown in Table 2, with the design activities every designer can perform, explained in Table 3.

On the left side of Figure 8, all design activities are represented graphically in order to visualize the working principle of the activity diagram. Here, the private workspace represents the area of responsibility of stakeholder X (see Figure 6).

Table 2. Decision vector

Abbreviation	Description	Var.
REQ AVAI	All requirements on the own quantities of interest are available in the public database.	α_1
REQ NEW	At least one requirement on the own quantities of interest in the public database is new.	α_2
REAL AVAI	All realizations of the own design variables are available in the public database.	α_3
REAL NEW	At least one realization of the own design variables in the public database is new.	α_4
REQ SATIS	All harmonized requirements on the own quantities of interest in the private workspace are satisfied.	α_5

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Table 3. Design activities

Abbreviation	Description	Pseudocode

COMP	Compare requirements on quantities of interest and realizations of design variables in private workspace with requirements on quantities of interest and realization of design variables in public database.	$\alpha_1 \leftarrow 1,\ \alpha_2 \leftarrow 0$
		for all $j \in \{1,\ldots,k\}$ **do**
		  if $[\mathbf{y}_{j,lb}^{PD} \notin \mathbb{R}] \vee [\mathbf{y}_{j,ub}^{PD} \notin \mathbb{R}]$
		  $\alpha_1 \leftarrow 0$
		  end
		  if $[\mathbf{y}_{j,lb}^{PW} \neq \mathbf{y}_{j,lb}^{PD}] \vee [\mathbf{y}_{j,ub}^{PW} \neq \mathbf{y}_{j,ub}^{PD}]$
		  $\alpha_2 \leftarrow 1$
		  end
		end
		$\alpha_3 \leftarrow 1,\ \alpha_4 \leftarrow 0$
		for all $i \in \{1,\ldots,h\}$ **do**
		  if $x_{i,r}^{PD} \notin \mathbb{R}$
		  $\alpha_3 \leftarrow 0$
		  end
		  if $x_{i,r}^{PD} \neq x_{i,r}^{PW}$
		  $\alpha_4 \leftarrow 1$
		  end
		end

PULL REQ ON QOI	Transfer requirements on quantities of interest from public database into private workspace.	**for all** $j \in \{1,\ldots,k\}$ **do**
		  $\mathbf{y}_{j,lb}^{PW} \leftarrow \mathbf{y}_{j,lb}^{PD}$
		  $\mathbf{y}_{j,ub}^{PW} \leftarrow \mathbf{y}_{j,ub}^{PD}$
		end

HARM REQ ON QOI	Harmonize requirements on quantities of interest in private workspace.	**for all** $j \in \{1,\ldots,k\}$ **do**
		  $y_{j,lb,H}^{PW} \leftarrow \max\{\mathbf{y}_{j,lb}^{PW}\}$
		  $y_{j,ub,H}^{PW} \leftarrow \min\{\mathbf{y}_{j,ub}^{PW}\}$
		  if $y_{j,ub,H}^{PW} \leq y_{j,lb,H}^{PW}$
		  $y_{j,lb,H}^{PW} \leftarrow \overline{\frac{1}{2}\left[\mathbf{y}_{j,lb}^{PW} + \mathbf{y}_{j,ub}^{PW}\right]}$
		  $y_{j,ub,H}^{PW} \leftarrow \overline{\frac{1}{2}\left[\mathbf{y}_{j,lb}^{PW} + \mathbf{y}_{j,ub}^{PW}\right]}$
		  end
		end

PULL REAL OF DV	Transfer realizations of design variables from public database into private workspace.	**for all** $i \in \{1,\ldots,h\}$ **do** $x_{i,r}^{PW} \leftarrow x_{i,r}^{PD}$ **end**
QUANT REAL OF QOI	Quantify realizations of quantities of interest in private workspace based on realizations of design variables in private workspace.	**for all** $j \in \{1,\ldots,k\}$ **do** $y_{j,r}^{PW} \leftarrow f\left(x_{1,r}^{PW}, x_{2,r}^{PW}, \ldots, x_{h,r}^{PW}\right)$ **end**
PUSH REAL OF QOI	Transfer realizations of quantities of interest from private workspace into public database.	**for all** $j \in \{1,\ldots,k\}$ **do** $y_{j,r}^{PD} \leftarrow y_{j,r}^{PW}$ **end**
EVAL REQ ON QOI	Evaluate fulfillment of harmonized requirements on quantities of interest in private workspace based on realizations of quantities of	

Continued

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Table 3. Continued

Abbreviation	Description	Pseudocode

	interest in private workspace.	$\alpha_5 \leftarrow 1$
		for all $j \in \{1, ..., k\}$ **do**
		  if $[\, y_{j,r}^{PW} \leq y_{j,lb,H}^{PW} \,] \vee [\, y_{j,ub,H}^{PW} \leq y_{j,r} \,]$
		  $\alpha_5 \leftarrow 0$
		  end
		end

DERIVE REQ ON DV	Derive requirements on design variables in private workspace based on harmonized requirements on quantities of interest in private workspace.	$\min_{\xi} f(\xi)$ subject to: $h(\xi) = 0$ $g(\xi) \leq 0$ $\xi_l \leq \xi \leq \xi_u$

PUSH REQ ON DV	Transfer requirements on design variables from private workspace into public database.	**for all** $i \in \{1, ..., h\}$ **do**
		  $x_{i,lb}^{PD} \leftarrow x_{i,lb}^{PW}$
		  $x_{i,ub}^{PD} \leftarrow x_{i,ub}^{PW}$
		end

WAIT	Wait (no action).	

Each designer's overall *objective* is to satisfy the requirements received on the own quantities of interest by specifying requirements on the own design variables as the realizations of both of them cannot be manipulated directly but do result from what is returned from below. Formally (in terms of the given notation), this means specifying all $x_{i,lb}^{PW}$ and $x_{i,ub}^{PW}$ such that all $x_{i,r}^{PW}$ returned from below satisfy

$$y_{j,lb,H}^{PW} \leq y_{j,r}^{PW} \leq y_{j,ub,H}^{PW} \quad \forall j \in \{1,\ldots,k\}, \tag{2}$$

whenever

$$x_{i,lb}^{PW} \leq x_{i,r}^{PW} \leq x_{i,ub}^{PW} \quad \forall i \in \{1,\ldots,h\}. \tag{3}$$

Performing design activities, like deriving requirements, quantifying realizations, transferring design information, or, evaluating requirements, helps a designer to assess and achieve that overall design goal. Decision-making in terms of selecting and executing certain design activities is formalized in the UML activity diagram which we propose in this paper. Its working principle can be explained as follows: by default, a designer performs WAIT which means that nothing is done until the next activation takes place. Only to initialize the model right at the beginning of a simulation the START position is used. If activated a designer transitions from WAIT to COMP and then, depending on the current status of its decision vector, advances further through the activity diagram until the WAIT

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

position is reached again. Designers can perform multiple design activities during a single activation or a single design activity for a period of multiple activations.

To account for different types of behaviour the activity diagram has two modes: Mode A and Mode B. In Mode A new requirements are only broken down into requirements on the next lower level if the existing realizations of the quantities of interest do not satisfy them. In Mode B, however, new requirements are always broken down independent of whether they are satisfied by the current realizations or not. This is important to model as the individual decision-making of stakeholders may affect the dynamics of design processes.

To prevent the order of designers from impacting the simulation results, the time-discrete routine is defined as a two-stage process in each iteration: first, all designers perform their design activities which do not include any transfer of design information from the private workspace into the public database. Then, in the second phase, all designers perform their remaining design activities.

After the second phase, a subroutine transforms all the requirements assigned to the attributes at the bottom of the ADG in the public database into realizations. This is done by first determining the average between all lower and upper bounds that are linked to an attribute and then selecting the centre of those two limits.

Top-down requirement decomposition

Deriving requirements on design variables during DERIVE REQ ON DV is done by solving an optimization problem with the design variables $\boldsymbol{\xi}$ and the objective function $f(\boldsymbol{\xi})$ subject to specific constraints. The individual formulae depend on two criteria: first, whether requirements on design variables shall be specified as points or intervals, and, second, whether realizations of design variables returned from below, so-called *bottom-up information*, are available and shall be taken into account. After an optimization, the final results ($\boldsymbol{\xi}_f$) are turned into requirements on the design variables in the private workspace.

Figure 9 shows point-based (a) and interval-based (b) requirement decomposition with and without bottom-up information in a design scenario stakeholder X could be in. The green areas represent *good designs* (combinations of x_1 and x_2 which satisfy the requirements on y_1 and y_2). The red areas represent *bad designs* (combinations of x_1 and x_2 which do not satisfy the requirements on y_1 and y_2). A set of good designs is denoted as *solution space*. The set of all good designs is called a *complete solution space* (Zimmermann & von Hoessle 2013).

Point-based ($\boldsymbol{\xi} = \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{h-1}, x_h$): When formulating point-based requirements on design variables without bottom-up information available the objective is to minimize $f(\boldsymbol{\xi}) = \max_j \varphi_{TD_j}$ with

$$\varphi_{TD_j} = \max_j \left\{ \frac{\tilde{y}_j(\boldsymbol{\xi}) - y^{PW}_{j,ub,H}}{y^{PW}_{j,ub,H}}, \frac{y^{PW}_{j,lb,H} - \tilde{y}_j(\boldsymbol{\xi})}{y^{PW}_{j,lb,H}} \right\}. \tag{4}$$

In Figure 9(a), this is shown as point A. This approach also leads to requirements on design variables if there is no solution space, i.e., no design at all that would fulfil the requirements on y_1 and y_2.

In the following, assume that for some realizations of x_1 and x_2, see point B in Figure 9(a), the realizations of y_1 and y_2 do not satisfy the given requirements. This

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

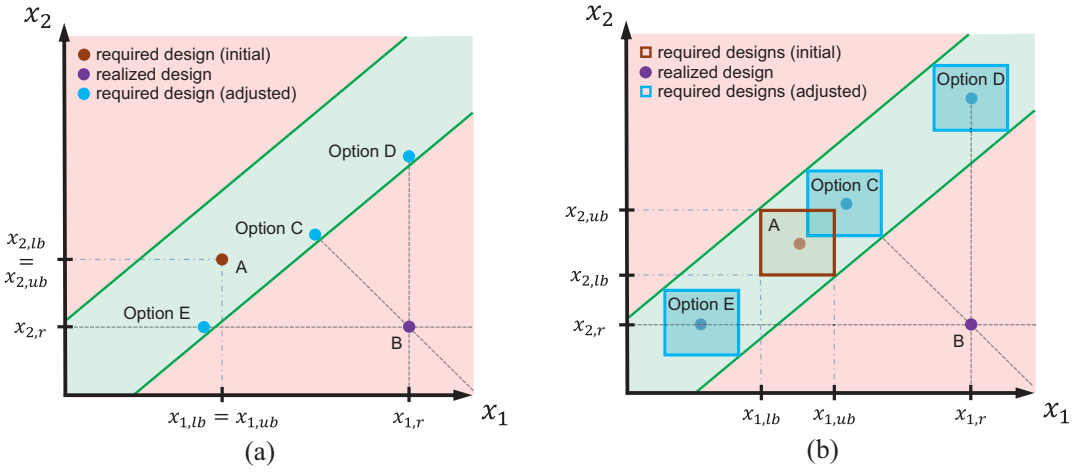


Figure 9. Point-based requirement decomposition (a) and interval-based requirement decomposition (b) with and without bottom-up information. The green areas represent all good designs. The red areas represent all bad designs.

could be due to some constraints on the component level (on t_1 and t_2), like material or manufacturing restrictions, that stakeholder X is unaware of.

In situations like that, we assume that designers redefine the requirements of their design variables considering the realizations of the design variables returned from below. This could be done in two ways: either by moving the new required design as close as possible to the realized design while still making sure that the new required design is good. In Figure 9(a), this is depicted as point C or by using the realization of one design variable as a new requirement and then adjusting the requirements on all other design variables such that the distance to the realized design is minimized while still ensuring that the new required design is good. In Figure 9(a) this is shown as points D and E (one is chosen). We will call the approach leading to point C Strategy 1 and to point D or E Strategy 2.

When formulating point-based requirements on design variables with bottom-up information according to Strategy 1 or Strategy 2 the objective is to minimize $f(\boldsymbol{\xi}) = \max_i \varphi_{BU_i}$ with

$$\varphi_{BU_i} = \gamma_i \frac{|\tilde{x}_i - x_{i,r}^{PW}|}{x_{i,r}^{PW}}, \tag{5}$$

subject to $f(\xi) = \max_j \varphi_{TD_j} \leq 0$ (The new required design shall be a good design).

The values assigned to γ_i (weighting factor for each design variable) determine which strategy is applied. In the case of Strategy 1, γ_i is 1, if the associated design variable is a quantity of interest in an area of responsibility below, or, a design variable in an area of responsibility on the same hierarchical level. Otherwise γ_i is 0, i.e., for design variables (on the bottom level) where no other stakeholder is assigned to the realizations do not have to be considered in Eq. (5). In the case of Strategy 2, we propose a metric that compares the old required design with the given realized design in each dimension to evaluate which realization is closest to

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

the specified requirement ($|x_{i,r}^{PW} - \frac{1}{2}\left(x_{i,lb}^{PW} + x_{i,ub}^{PW}\right)|/x_{i,r}^{PW}$). The realization of that design variable (where the metric is lowest) is then accepted as a new requirement. Numerically this means using the same assignment rule for γ_i as in the case of Strategy 1 and then scaling the weighting factor γ_i of the selected design variable disproportionately high.

Interval-based ($\boldsymbol{\xi} = \tilde{x}_{1,lb}, \tilde{x}_{2,lb}, \ldots, \tilde{x}_{(h-1),ub}, \tilde{x}_{h,ub}$): When formulating interval-based requirements on design variables without bottom-up information available the objective is to minimize $f(\boldsymbol{\xi}) = -\mu(\Omega)$ where Ω is a *solution box*, a subspace of the complete solution space that is only allowed to include good designs

$$\Omega = [\tilde{x}_{1,lb}, \tilde{x}_{1,ub}] \times \ldots \times [\tilde{x}_{h,lb}, \tilde{x}_{h,ub}], \tag{6}$$

and $\mu(\Omega)$ is a size measure that is used to determine the suitability of a solution box (Zimmermann & von Hoessle 2013; Zimmermann *et al.* 2017). A typical measure, for example, is the volume of a solution box (used in this study)

$$\mu(\Omega) = (\tilde{x}_{1,ub} - \tilde{x}_{1,lb})(\tilde{x}_{2,ub} - \tilde{x}_{2,lb}) \ldots (\tilde{x}_{h,ub} - \tilde{x}_{h,lb}). \tag{7}$$

In Figure 9(b), the initial solution box is denoted as A. The projection of its edges onto the coordinate axis shows the lower and upper boundary of the requirement assigned to each design variable.

Now, assume that for some realizations of x_1 and x_2, see point B in Figure 9(b), the realizations of y_1 and y_2 do not satisfy the requirements. In this situation, designers may react similarly as before, i.e., by redefining the requirements on their design variables using the bottom-up information according to Strategy 1 or 2 in order to make it easier for the recipients of the requirements to satisfy them. This time, however, the centre of the box is used as a point of reference. With respect to Strategy 1, this causes a conflict of goals between moving the centre of the box as close as possible to the realized design and maximizing the size of the box, as shown by box C in Figure 9(b). Thus, Strategy 1 is not considered further. In the case of Strategy 2 designers rearrange the requirements on their design variables such that the centre of the new box is in line with the realization which is closest to the centre of the old box while still maximizing the size of the new box. In Figure 9(b) this is shown by boxes D and E.

When deriving interval-based requirements on design variables with bottom-up information according to Strategy 2, the objective is to minimize $f(\boldsymbol{\xi}) = -\mu(\Omega)$ subject to

$$|x_{i,r}^{PW} - \frac{1}{2}(\tilde{x}_{i,lb} + \tilde{x}_{i,ub})| = 0 \tag{8}$$

for the design variable with the highest value of $|x_{i,r}^{PW} - \frac{1}{2}\left(x_{i,lb}^{PW} + x_{i,ub}^{PW}\right)|/x_{i,r}^{PW}$. In Figure 9(b) this results in one of the solution boxes, D or E.

If an optimization fails to identify point- or interval-based requirements under consideration of bottom-up information (e.g. because there is no solution space), a second optimization step is carried out in order to find point-based requirements without bottom-up information (which always yields a solution).

Note that all approaches for requirement (re)formulation that we suggest also work in high dimensions and for arbitrary solution spaces.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Example problem

In the following section, our model is applied to a simple design problem from the field of vehicle development. This allows us to illustrate the model behaviour and to examine the effect of point-based and interval-based requirement decomposition under consideration of different types of design behaviour. While understanding the model behaviour is essential for the validation of the model, the evaluation of interval-based requirement decomposition helps to assess SSE.

Simplified vehicle development problem

The example problem at hand is a simplified vehicle design task with six product properties and three designers which are called Accelerator (ACC), Mass Master (MM) and Engineer (ENG). Figure 10 shows the corresponding ADG and Table 4 lists the responsibilities and boundaries of the design space.

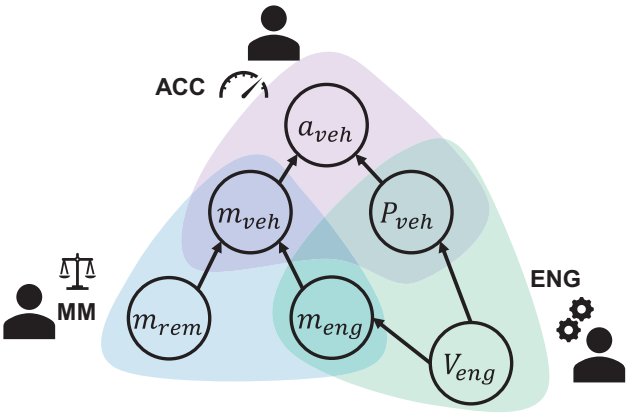


Figure 10. ADG of the example problem.

Table 4. Product properties of the example problem, the limits of the design space and the assignment to designers as quantities of interest (QoI) and design variable (DV)

Property	Description	Min.	Max.	ACC	MM	ENG
a_{veh} $\left[\frac{\mathrm{m}}{\mathrm{s}^2}\right]$	Acceleration of the vehicle	0	7	QoI	-	-
P_{veh} [kW]	Power of the vehicle	0	150	DV	-	QoI
m_{veh} [kg]	Total mass of the vehicle	0	2250	DV	QoI	-
m_{rem} [kg]	Remaining mass of the vehicle	1500	1500	-	DV	-
m_{eng} [kg]	Mass of the engine	0	750	-	DV	QoI
V_{eng} [L]	Displacement vol. of the engine	0	3	-	-	DV

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Overall, the objective is to fulfil a requirement assigned to the acceleration of the vehicle (a_{veh}) by identifying an appropriate engine design (realization of the displacement volume of the engine, V_{eng}). It is assumed that the remaining mass of the vehicle (m_{rem}) is fixed at 1500.0 kg.

The acceleration of the vehicle was measured at a speed of $v_0 = 10\frac{\mathrm{m}}{\mathrm{s}}$ is defined as

$$a_{veh} = \frac{P_{veh}}{v_0 m_{veh}}, \tag{9}$$

with the power of the vehicle denoted as P_{veh} and the (total) mass of the vehicle denoted as m_{veh}. The total mass of the vehicle depends on the mass of the engine (m_{eng}) and the remaining mass of the vehicle (m_{rem})

$$m_{veh} = m_{eng} + m_{rem}. \tag{10}$$

It is assumed that both, the mass of the engine and the power of the vehicle are functions of the displacement volume of the engine

$$m_{eng} = \eta V_{eng} \text{ with } \eta = 250\frac{\mathrm{kg}}{\mathrm{L}}, \tag{11}$$

and

$$P_{veh} = \kappa V_{eng} \text{ with } \kappa = 50\frac{\mathrm{kW}}{\mathrm{L}}. \tag{12}$$

Step-by-step analysis of model behaviour

Figure 11 illustrates the first ten iterations of a simulation. An ADG visualizes the flow of design information (left) and a representation of the input or output space on the associated level depicts the evolution of all requirements and realizations (right). The green line consists of all *feasible* designs. Here, feasible means that the associated designs could be realized by choosing an appropriate realization of the displacement volume of the engine. Feasibility is limited - each combination of vehicle mass and vehicle power requires a unique displacement volume of the engine.

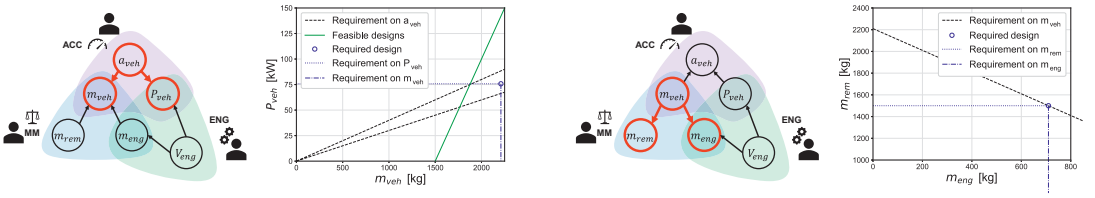
We assume that designers operate according to Mode A (see activity diagram). Further, it is assumed that designers formulate point-based requirements according to Strategy 2. The design task is to satisfy the following requirement

$$3\frac{\mathrm{m}}{\mathrm{s}^2} \leq a_{veh} \leq 4\frac{\mathrm{m}}{\mathrm{s}^2}. \tag{13}$$

No further requirements or realizations are specified. It is assumed that only the decomposition of requirements and the quantification of realizations require one iteration to complete. All other design activities happen instantaneously.

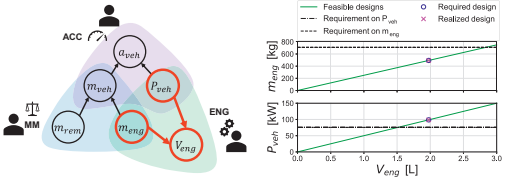
At the beginning of the simulated design process, during iterations one, two and three, design information propagates top-down as the requirement assigned to the acceleration of the vehicle, specified in Eq. (13), is repeatedly decomposed into requirements on lower levels until the bottom of the ADG (V_{eng}) is reached. This includes the Accelerator which breaks down the desired acceleration of the vehicle into a new requirement on the total mass of the vehicle (2209.8 kg) and the power

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

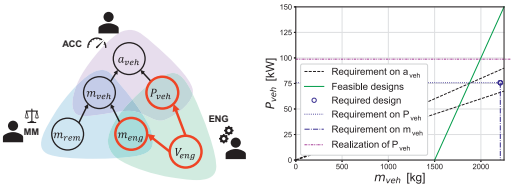


(a) Iteration 1

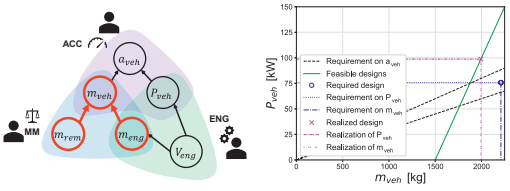
(b) Iteration 2



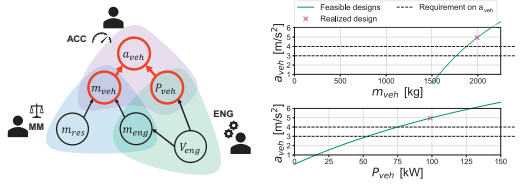
(c) Iteration 3



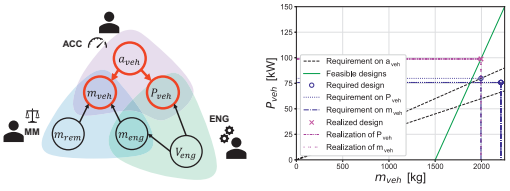
(d) Iteration 4



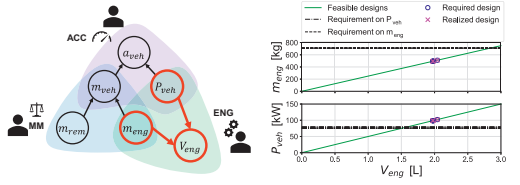
(e) Iteration 5



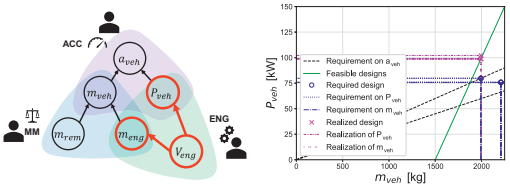
(f) Iteration 6



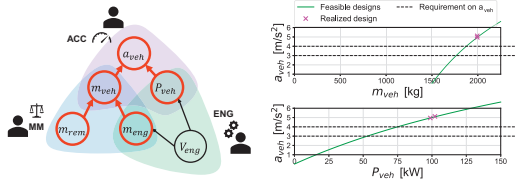
(g) Iteration 7



(h) Iteration 8



(i) Iteration 9



(j) Iteration 10

Figure 11. Flow of design information during the first ten iterations of a simulation (Mode A, Strategy 2).

of the vehicle (75.8 kW), the Mass Master who then uses the requirement on the total mass of the vehicle to formulate a requirement on the remaining mass of the vehicle (1500.0 kg) and the mass of the engine (709.8 kg), and, the Engineer who derives a requirement on the displacement volume of the engine (1.98 L) based on

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

the requirement on the mass of the engine and the power of the vehicle. Note that due to the limitations on feasibility mentioned before the Engineer has to resolve a *conflict of goals*. He or she can either choose a large engine which would yield a proper mass but also too much power, or, a small engine with appropriate power but too little mass.

After the requirements on the remaining mass of the vehicle and the displacement volume of the engine are turned into realizations (1.98 L; 1500.0 kg) design information propagates back up during iterations four, five and six as each designer quantifies the realizations of its quantity of interest based on the feedback from below. This includes the Engineer who determines the realization of the mass of the engine (494.0 kg) and the power of the vehicle (98.8 kW) based on the realization linked to the displacement volume of the engine, the Mass Master who then quantifies the realization of the total mass of the vehicle (1994.0 kg) by using the realization linked to the mass of the engine and the remaining mass of the vehicle, and finally, the Accelerator determines the realization of the acceleration of the vehicle ($4.9\frac{\mathrm{m}}{\mathrm{s}^2}$) based on the realization of the total mass and the power of the vehicle. Afterwards, the Accelerator realizes that the requirement assigned to the acceleration of the vehicle is not satisfied by the current realization ($a_{H,ub}^{PW} < a_r^{PW}$).

During iteration seven the Accelerator then redefines the requirements on the total mass of the vehicle (1994.0 kg) and the power of the vehicle (79.8 kW) such that the new requirement on the total mass matches the corresponding realization. This forces the Engineer to redefine its requirement on the displacement volume of the engine (2.04 L) during iteration eight while the Mass Master stays inactive (as the requirement on the total mass of the vehicle is satisfied by the realization he or she currently provides). Note that, in this situation, the Engineer formulates a requirement on its design variable based on a requirement assigned to the power of the vehicle (specified by the Accelerator in iteration seven), and, a requirement assigned to the mass of the engine (specified by the Mass Master in iteration two). Thus, the requirements linked to the quantity of interest of the Engineer (in iteration eight) originate from two different sets of requirements defined by the Accelerator (iterations one and seven). We call this an *inconsistent* design.

In the following, after the new requirement on the displacement volume of the engine is turned into a realization (2.04 L), design information propagates back up once again. This includes the Engineer who, during iteration nine, quantifies the realization linked to the mass of the engine (510.7 kg) and the power of the vehicle (102.1 kW) based on the realization of the displacement volume of the engine, and, the Mass Master and the Accelerator who, during iteration ten, use the realization linked to the mass of the engine and the power of the vehicle in order to determine the realization of the total mass of the vehicle (2010.7 kg) and the acceleration of the vehicle ($5.1\frac{\mathrm{m}}{\mathrm{s}^2}$). In this situation, the Accelerator quantifies a realization of its quantity of interest based on a realization of the power of the vehicle (defined by the Engineer in iteration nine), and, a realization of the total mass of the vehicle (defined by the Mass Master in iteration five). Thus, the realizations linked to the design variables of the Accelerator (in iteration ten) originate from two different realizations the Engineer has assigned to the displacement volume of the engine (iterations nine and four). We call this an *inconsistent* design as well.

In summary, the first ten iterations of the simulation reveal three results: first, local design iterations, i.e., repetitions of design activities, arise naturally without

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

being predefined (designers are not instructed to repeat design activities). Second, the flow of design information alternates between the top and bottom of the ADG. Third, inconsistent designs occur. The requirements and realizations available to designers at some point in time are *inconsistent* if they originate from at least two different sets of requirements or realizations another designer above or below has assigned to its properties.

Study of point- vs. interval-based design

In order to systematically evaluate point- and interval-based requirement decomposition under consideration of different types of design behaviour four different simulations are carried out. Each consists of 100 iterations. The setup of the study is shown in Table 5. We assume that only the decomposition of requirements and the quantification of realizations require one iteration to complete.

Figure 12 shows the results of the study including the evolution of all required and realized designs on the subsystem level (left side), and, the evolution of the requirement and realization assigned to the acceleration of the vehicle (right side).

Note that the required design on the far right side of Figure 12(a), (c), (e) and (g) shows the first required design provided by the Accelerator.

Point-based design

First, consider Mode A. When evaluating Figure 12(a) note that the required and realized designs (gradually) converge towards the area where all feasible engines satisfy the system-level requirement on the acceleration of the vehicle. It seems as if it takes a couple of iterations to identify an engine design that is both: feasible and good. Figure 12(b) confirms this observation from a system-level viewpoint. Sometimes the distance between the requirement and the realization linked to the acceleration of the vehicle also increases as the Accelerator uses an inconsistent (realized) design to determine the realization of the acceleration of the vehicle.

Now, observe Mode B. Here, the evolution of all required and realized designs on the subsystem level shown in Figure 12(c) is similar. This time, however, less iterations are needed to find a suitable realization of the displacement volume of the engine. Figure 12(d) confirms this.

Interval-based design

First, consider Mode A. Figure 12(e) shows that the required and realized designs also move towards the area where all feasible engines satisfy the design target on the acceleration of the vehicle. The requirements on the total mass and the power of the

Table 5. Setup of the simulation study. Requirements are always reformulated according to Strategy 2

	Simulation #1	Simulation #2	Simulation #3	Simulation #4
Requirements	Point-based	Point-based	Interval-based	Interval-based
Activity diagram	Mode A	Mode B	Mode A	Mode B
Figure 12	(a), (b)	(c), (d)	(e), (f)	(g), (h)

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

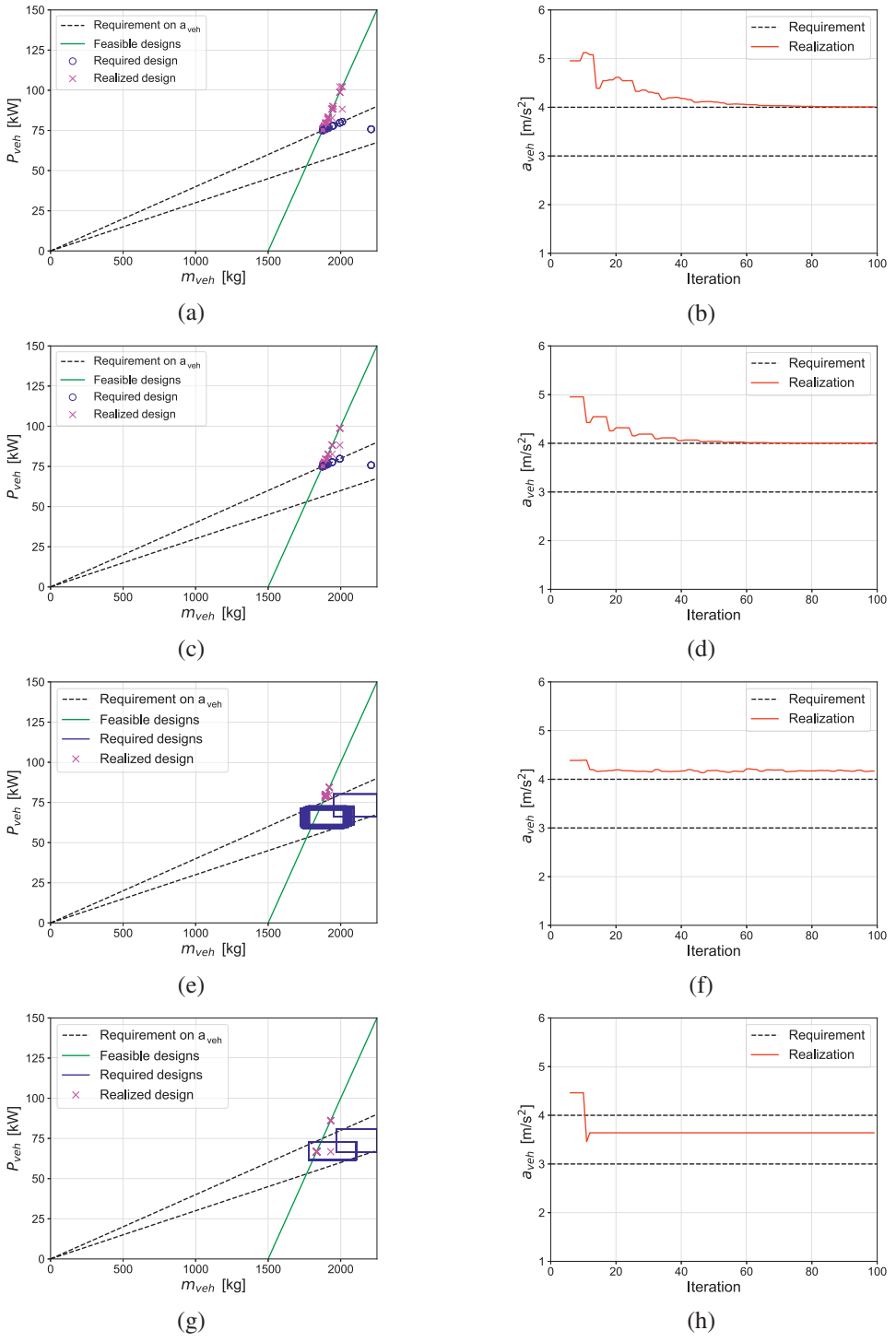


Figure 12. Point-based (a, b, c, d) and interval-based (e, f, g, h) requirement decomposition for Mode A (a, b, e, f) and Mode B (c, d, g, h) according to Strategy 2.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

vehicle are constantly redefined so that the centre of the solution box (blue) is in line with the realization returned for the mass of the vehicle according to Eq. (8). This happens because the realization returned for the total mass of the vehicle is closer to the associated requirement than the realization returned for the power of the vehicle. Yet even though multiple iterations take place no suitable engine design can be found. The realization linked to the acceleration of the vehicle does not fall below the upper limit of the specified requirement, see Figure 12(f). This *Deadlock* is caused by the Mass Master who (at some point) does not update its requirements and, thus, blocks the flow of design information from the top to the bottom of the ADG. The underlying mechanism is explained below.

Deadlock: first, the Mass Master receives a requirement on the total mass of the vehicle ($1952.0\,\mathrm{kg} \leq m_{veh} \leq 2250\,\mathrm{kg}$) and breaks this down into a requirement on the remaining mass of the vehicle ($1500.0\,\mathrm{kg} \leq m_{rem} \leq 1500.0\,\mathrm{kg}$) and the mass of the engine ($444.1\,\mathrm{kg} \leq m_{eng} \leq 750.0\,\mathrm{kg}$). After the requirement on the remaining mass is turned into a realization (1500.0 kg), and, a realization linked to the mass of the engine is reported back (421.5 kg), the resulting realization of the total mass of the vehicle (1921.5 kg) is transferred upwards. Then, the Mass Master receives a new requirement on the total mass of the vehicle ($1175.9\,\mathrm{kg} \leq m_{veh} \leq 2092.8\,\mathrm{kg}$), which, however, is already satisfied by the current realization. As a consequence, he or she does not propagate the new design information down, even though, the Engineer depends on it. Note that a minor change of the requirement assigned to the total mass of the vehicle or of the realization linked to the mass of the engine does not force the Mass Master to reformulate its requirements since the interval-based requirement on its quantity of interest provides much room to be satisfied. In case the requirement on the total mass of the vehicle is formulated as a point, however, the Mass Master reformulates its requirements if the received requirement, or, the received realizations change slightly as point-based requirements on quantity of interest only provide little room to be satisfied.

Now, consider Mode B. Figure 12(g) reveals that the evolution of all required and realized designs on the subsystem level is similar to Mode A. Here, however, a suitable engine design is found after ten iterations. The realized design quickly reaches the area where feasibility is given and all design goals are fulfilled. Figure 12(h) confirms this observation as the realization linked to the acceleration of the vehicle instantly jumps below the upper limit of the associated requirement.

Note that in both cases of interval-based requirement formulation inconsistent designs occur as well.

Discussion

The simulation model developed in this work and the results of the computational study are discussed in five ways: first, the simulation results are interpreted and relevant mechanisms are identified. Then, the strengths of the model are outlined and information about the validation is provided. In the end, the limitations of this study are stated and the implications for science and industry are described.

Interpretation of results

Our results show that development processes with agent behaviour as specified in our model are characterized by iterations if requirements are formulated as target

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

points. Iterations create knowledge and provide progress. Yet, they also increase cycle time and cost (Wynn & Eckert 2017). Our results reveal one major cause for iterations: conflict of goals that emerge when stakeholders on lower levels try to satisfy several requirements (which are contradictory) with a limited number of design variables. This is especially noticeable for requirements that are defined as target points as they only provide little design freedom and represent a challenge to be satisfied. The subsequent design trade-offs then force the stakeholders above to adjust the associated requirements. This interplay between seeking design trade-offs and reformulating requirements only allows a slow and gradual improvement of the realized design on the system level. As an example, for this consider simulation #1 and #2. Here, the Engineer continuously receives two conflicting requirements that he or she cannot fulfill simultaneously. The point-based design targets simultaneously demand a high weight and a low power of the engine which is not feasible. The trade-offs made by the Engineer then force the Mass Master and the Accelerator above to reformulate their requirements. Those, however, necessitate further trade-offs from the Engineer during subsequent design iterations since the new target points can, once again, not be reached simultaneously due to the feasibility constraints on the engine design.

One way to reduce the number of iterations is to directly decompose any new requirements received from stakeholders above instead of first evaluating whether they might already be fulfilled (by the current realizations) or not. The reason for this is that each subsystem quantification and evaluation require additional time while most often (especially during point-based design) the new requirements are broken down afterward anyway (because during point-based design it is unlikely that the current realizations exactly satisfy the current requirements). To witness this, compare simulation #1 and #2. Here, the realization linked to the acceleration of the vehicle approaches the requirement faster if the designers are instructed to decompose any new design goals immediately, see Figure 12(b) and (d). This is mainly due to the Mass Master who saves time by not quantifying and evaluating the own subsystem each time new requirements and new realizations are present. Instead, he or she instantly decomposes the new requirement on the total mass of the vehicle (as defined in Mode B) and therefore accelerates the transfer of design information from the Accelerator to the Engineer.

A second way to reduce iterations and therefore cut cycle time is to formulate requirements on design variables as target intervals instead of as target points, i.e., interval-based instead of point-based. According to our results, this significantly decreases the time necessary to find an appropriate design in the case of Mode B, i.e., when new requirements are directly broken down. To observe this effect compare simulation #2 and #4. The significant drop in development time, see Figure 12(d) and (h), is due to the large amount of design information that propagates top-down from the Accelerator to the Engineer who then has a better chance to find a suitable component design as there are no conflicting requirements. Formulating design targets as ranges instead of as points can resolve conflicts of goals (Königs & Zimmermann 2016) and therefore accelerate development processes.

The existing results, however, also show that if updates of design goals are not directly broken down but first evaluated with respect to whether they are already satisfied or not (see Mode A) an interval-based formulation of design targets can also make development processes stall. This means that no satisfying component

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

design is identified at all. This effect can, for example, be observed in simulation #3. Here, the design process enters what we call a deadlock caused by the actions of the Mass Master. At some point, he or she does not update the requirements on the remaining mass of the vehicle and the mass of the engine because the current realizations assigned to those properties already fulfill the design goal on the total mass of the vehicle (specified by the Accelerator as target range). As a result, critical design information does not propagate top-down and the Engineer on the bottom level is unaware of what component performance is required to satisfy the design goal on the top level. This mechanism is driven by the definition of design goals as ranges instead of as points as those provide much room to be satisfied and thus can remove the incentive for subsystem designers to reformulate their own design goals. If design goals are formulated as target points even small modifications of the requirements and realizations received from above and below force subsystem designers to update their own requirements. This can, for example, be witnessed in simulations #1 and #2.

The results of our work also shed light on how inconsistent design information emerges and what it causes. An example of this is shown in simulation #1. Here, the Engineer and the Accelerator sometimes formulate requirements and quantify realizations based on design information available to them which originates from two different required and realized designs a stakeholder above or below has specified, see iterations eight and ten. Two factors drive the emergence of inconsistent design information: the structure of an ADG and the distribution of responsibility. The combination of both determines how long design information needs to travel up and down an ADG. If, for example, ADGs are horizontally asymmetrical, like in Figure 10, and stakeholders are located on *side paths*, like the Mass Master, the requirements and realizations on those side paths need more time to be transferred up and down than elsewhere. This can, for example, be observed during iteration ten of simulation #1. Here, the Mass Master still quantifies the realization of the mass of the engine defined by the Engineer in iteration nine, while the Accelerator already quantifies the realization linked to the power of the engine defined by the Engineer in iteration nine combined with the realization assigned to the total mass of the vehicle specified by the Mass Master in iteration five, see Figure 11. Thus, the most recent realization of the total mass of the vehicle has not yet reached the top as the Mass Master is still processing it.

It needs to be mentioned that inconsistent design information as defined by us can also emerge in the case of horizontally symmetrical ADGs. If, for example, some designers take more time to derive requirements or quantify realizations the flow of design information is delayed as well. This, however, cannot be witnessed in our simulations as all of the designers in our study need the same amount of time to complete the given design activities. Note that we observe inconsistent design information independent of whether requirements are formulated as points, or, as intervals. This can be seen when comparing simulations #1 and #3. Moreover, the agent behaviour does not seem to influence the emergence of inconsistent design information. To observe this, compare #1 and #2.

The emergence and propagation of inconsistent design information have some serious consequences. Our results highlight two issues: first, realized designs on the system level are quantified based on essentially wrong design information on the subsystem level. This provides a false impression of the current performance of a product. As an example of this, consider iteration ten in simulation #1. Here, the

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Accelerator quantifies a realization of the acceleration of the vehicle based on a realization of the total mass and the power of the vehicle which in combination represent an unfeasible design. A second issue is that components on the bottom level are developed based on misleading requirements. If those faulty designs are propagated up again the stakeholders on the top receive designs that are different from what they have requested. Furthermore, it causes unnecessary development work. This phenomenon can, for example, be seen in iteration eight of simulation #1. Here, the Engineer specifies a requirement on the displacement volume of the engine based on a requirement assigned to the power of the vehicle and the mass of the engine which in combination represent a design that has not been requested as such by the Accelerator. Note that designers are unaware of whether the design information available to them is inconsistent due to their limited scope.

Strengths of the model

The results of the computational study reveal that the model we propose has three strengths (besides the originally intended strength to simulate product-related and organizational hierarchy in a generic way): convergence behaviour, the occurrence of design iterations and the emergence of inconsistent design information.

Convergence: The designers in our model manage to identify acceptable and feasible product designs within the given period of time when assuming that they act according to Mode B and Strategy 2. This means they converge to a solution. Evidence for this can, for example, be found in simulations #1, #2 and #4. Note that convergence is not guaranteed. Multi-level optimization techniques, such as Analytical Target Cascading, for example, usually suffer from convergence issues (Michelena *et al.* 2003). To solve this issue, they usually incorporate sophisticated subroutines (Michalek & Papalambros 2004). We do not claim that our model always converges.

Design iterations: The designers in our model often repeat design activities multiple times in order to fulfill their design targets. Compared to other popular simulation models (Smith & Eppinger 1997a, 1997b) those design iterations emerge naturally meaning without being explicitly prescribed. This can, for example, be observed in simulations #1 and #2. The natural emergence of design iterations has two significant advantages: first, we can investigate their root causes, and second, we may test strategies to avoid or facilitate them.

Inconsistencies: The requirements and realizations available to the designers in our model can be inconsistent at one point, i.e., they originate from two or more different (sets of) requirements or realizations another designer (above or below) has assigned to its design variables or quantity of interest. To observe this effect, consider simulation #1 and #2 illustrated in Figures 11 and 12. Inconsistent design information is critical in real-world development processes where up to thousands of designers simultaneously exchange design information. Here, inconsistencies can occur on many hierarchical levels at the same time. This may cause confusion and ambiguity on what design is actually required and realized.

Validation of the model

We argue that the convergence behaviour, the emergence of design iterations, and the occurrence of inconsistent design information are important indicators for the

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

validity of our model as actual product design processes exhibit these phenomena as well. This type of validation technique is called *face validation* (Sargent 1992; Klügl 2008). It is often used for agent-based models that are difficult to validate in general due to the lack of empirical data.

Furthermore, the findings of our study on the influence of point- and interval-based requirement decomposition are similar to the conclusions drawn by Panchal *et al.* (2005) and (2007). In their work, the authors analyse the so-called *Interval-based Constraint Satisfaction* (IBCS) method by using a game-theoretic approach and non-cooperative agents. The IBCS method, which is based on SBD, assumes that stakeholders forward feasible ranges of values for design variables to other stakeholders instead of communicating single points in the design space. The authors state that this preserves design freedom and addresses the problem of unanticipated design iterations. Similar benefits can be witnessed in our study.

Limitations of this work

The results of this work are subject to limitations. In the following, we distinguish between constraints on the *validity* and *applicability* of our model and constraints on the *generalizability* of our findings.

Validity: Many factors could limit the ability of our model to represent actual design processes. One factor is the assumption that stakeholders only perform the given design activities while reality provides an incomplete number of activities that are executed in order to drive a product towards its desired design objectives. We, for example, do not consider any direct communication between stakeholders (e.g. conversation, message, meeting) or elicitation of system-level requirements in early design phases. Modelling requirement elicitation explicitly (via customer surveys, prototyping, benchmarking) could lead to different design goals assigned to the top-level product properties. Further, it is questionable whether the activity diagram that we introduce accurately describes the decision-making of individual stakeholders. Human design behaviour can also be irrational, random and biased. Important factors related to this could also be learning, skill, motivation, expertise and prior knowledge. All of them are neglected in our model. Compared to other studies we also ignore team dynamics and any benefits or drawbacks that occur if stakeholders derive design decisions in groups (instead of alone). Moreover, we assume that ADGs are *static*. This means that they do not change over the course of a development process. In reality, however, they evolve (they grow and become smaller) as components are added and removed. Some components might also be reused in other ADGs as companies sometimes develop multiple products in parallel that share cost-intensive parts. With respect to our model, this would mean that realizations assigned to certain product properties in one ADG must be identical to realizations assigned to product properties in other ADGs if those sets of properties represent the same component. A further limitation of our approach is the assumption that requirements are part of the process model. While this is in line with the V-model other process descriptions such as the one from Ulrich and Eppinger (2016), for example, do not explicitly regard requirements as a dynamic component of product development processes. Finally, we might make a mistake by modelling product-related and organizational hierarchy with the same general type of interface on each level of decomposition. In actual product development processes the individual interfaces and the way they are handled may vary from

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

level to level. The interactions between stakeholders within an organization, for example, may be different from the interactions between stakeholders that are part of two different organizations. This case may occur if components are developed by suppliers who then receive requirements from a company.

Applicability: The application of our model may be limited for the following reasons: first, it might be impossible to construct an ADG if, for example, distinct design variables and quantity of interest cannot be named. Second, quantitative models may not be available. Third, the distribution of responsibility may not be specified with respect to product properties but rather with respect to components and functions.

Generalizability: We do not claim that our findings are generally valid as we only study a single design problem under specific (selected) boundary conditions. It is questionable whether the conclusions we draw also apply to practical design problems. Those are typically larger and more complex as shown by the example in Figure 5. The present study is not intended to make accurate predictions about this kind of application. It is, however, worth noting that even the small, simple and less realistic example problem we study reproduces some phenomena known from design practice. This is somewhat counterintuitive as one could assume that those phenomena only occur if large and complex design problems are examined. Therefore, we cautiously suspect that the effects we observe can also be witnessed if the ADG and the number of participating designers are scaled up.

Implication and relevance

Theory

The theoretical contribution of our research may be twofold: on the one hand, we provide a powerful tool to analyse hierarchical product design processes in great detail as we model product properties and responsibilities on a micro-level. And, on the other hand, we foster the understanding of key mechanisms in distributed design as we analyse the root causes of design iterations and inconsistent design information. Further, we demonstrate the benefits and shortcomings of point- and interval-based requirement formulation.

Practice

Efforts to optimize product design processes in the real world are usually focused on accelerating design tasks, reordering design tasks, or reorganizing design teams. However, we are aware of only one case where stakeholders are supposed to formulate requirements as intervals (instead of as points) in order to improve a development process. Our results may provide quantitative decision support for others who are interested in implementing this approach (SSE) as well. Especially the strengths and weaknesses of SSE revealed by our simulations can help companies make a better decision on when and where to apply the approach.

Conclusion

In this study, we introduce a quantitative model for the simulation of hierarchical product design processes. It combines a directed, hierarchical graph as a product model with agents that are responsible for certain areas of that graph. Top-down and bottom-up flow of design information emerges as autonomous agents (in their role as designers) assign requirements and realizations to product properties that

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

are located at the interfaces of technical responsibility. The software architecture (code structure) of the model is defined by a UML class diagram. The decision-making of the individual designers is described by a UML activity diagram. To account for point-based and interval-based requirement formulation we provide mathematical problem statements for each case. Available bottom-up information (returned realizations) is considered by using an adaptive mechanism.

Our model has five key strengths: first, it generates design iterations naturally (without being prescribed). This allows studies on their root causes and about the consequences. Second, it captures inconsistent design information - an important factor in real-world development processes. Third, it can be used to analyse how responsibility should be allocated in order to enhance distributed design. Fourth, it can be used to evaluate how the specification of requirements (point- or interval-based) affects the development process. Fifth, it allows us to study how the product itself (size, coupling, complexity) may influence the design process.

The application of our model to a simple design problem leads to three major results: first, point-based requirement decomposition usually causes many design iterations, leading to long development times. Second, formulating design targets as ranges instead of as single values can reduce iterations and lower cycle time considerably if updates of requirements are always directly broken down. If not, interval-based requirement decomposition can also cause development processes to stall. Third, decomposing updates of design goals directly instead of evaluating them first concerning whether they are already satisfied or not is always beneficial in terms of development time.

Outlook

Future research could have three goals: first, applying our model to other design problems in order to examine product-related issues (e.g. *how does the structure of ADGs affect the development time?*), organization-related issues (e.g. *how does the distribution of responsibility affect the development time?*), or, process-related issues (e.g. *how does the duration of activities affect the development time?*). In this context, it would also be interesting to compare our model to mathematically motivated approaches for decomposing and solving hierarchical design problems. Those could, for example, be taken from Wagner (1993) or Allison *et al.* (2009). Second, validating our model based on experimental data obtained in multi-actor studies or industrial case studies. Third, extend our model to account for alternative modelling assumptions (e.g. skill of agents) or more realistic boundary conditions (e.g. modular products). Here, it would also be interesting to consider coordinated integration processes (Wöhr *et al.* 2023) or different behaviours used to facilitate system-level coordination. Collopy (2019) and Collopy *et al.* (2020), for example, differentiate between authority-based and emphatic leadership.

Financial support

This research was funded by the BMW Group.

List Of variables

x_i Generic design variable i

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

$x_{i,lb}$	Lower bound of design variable i
$x_{i,ub}$	Upper bound of generic design variable i
$x_{i,r}$	Realization of generic design variable i
y_j	Generic quantity of interest j
$y_{j,lb}$	Lower bound of generic quantity of interest j
$y_{j,ub}$	Upper bound of generic quantity of interest j
$y_{j,ub,H}$	Harmonized upper bound of generic quantity of interest j
$y_{j,r}$	Realization of generic quantity of interest j
z_i	System-level quantity of interest i
t_i	Component-level design variable i
α_i	Decision variable i
SX	Stakeholder X
P_{veh}	Power of the vehicle
m_{veh}	Total mass of the vehicle
m_{rem}	Remaining mass of the vehicle
m_{eng}	Mass of the engine
V_{eng}	Displacement volume of the engine
$\boldsymbol{\xi}$	Design variable vector
$\boldsymbol{\xi}_{lb}$	Lower bounds of design variable vector
$\boldsymbol{\xi}_{ub}$	Lower bounds of design variable vector
$f(\boldsymbol{\xi})$	Objective function
$h(\boldsymbol{\xi})$	Equality constraint
$g(\boldsymbol{\xi})$	Inequality constraint
$\boldsymbol{\xi}_f$	Vector of final optimization result
φ_{TD_j}	Objective meta-function (top-down)
φ_{BU_i}	Objective meta-function (bottom-up)
$\tilde{y}_j$	Design variable i for optimization
$\tilde{x}_i$	Quantity of interest j for optimization
γ_i	Weighting factor i
$\boldsymbol{\Omega}$	Solution box
μ	Size measure

List of abbreviations

ABM	Agent-based modelling
ABS	Actor-based signposting
ACC	Accelerator
ADG	Attribute dependency graph
AMPERE	Agent model for planning and research of early design
ATC	Analytical target cascading
AVAI	Available
BU	Bottom-up
CASIUM	Complex system integrated utilities model
CISAT	Cognitively inspired simulated annealing teams
COMP	Compare
D	Deep
DAG	Directed acyclic graph
DiFS	Design information flow simulation
DV	Design variable

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

ENG	Engineer
EVAL	Evaluate
FDT	Functional dependence table
GT	Game theory
H	Harmonized
IBCS	Interval-based constraint satisfaction
IPT	Integrated product teams
KABOOM	An agent-based organizational optimization model
LB	Lower bound
MCPD	Mass-collaborative product development
MM	Mass master
NPD	New product development
PD	Public database
PW	Private workspace
QOI	Quantity of interest
QUANT	Quantify
REAL	Realization
REQ	Requirement
S	Shallow
SATIS	Satisfied
SBCE	Set-based concurrent engineering
SBD	Set-based design
SE	Systems engineering
SiFA	Single function agents
SSE	Solution space engineering
TD	Top-down
UB	Upper bound
UML	Unified modelling language
VDT	Virtual design team

References

Albers, A. & **Meboldt, M.** 2007 IPEMM – Integrated product development process management model based on systems engineering and systematic problem solving. In *Proceedings of the 16th International Conference on Engineering Design ICED 07, Paris, France, No. 537*. The Design Society

Albers, A., **Reiss, N.**, **Bursac, N.** & **Richter T.** 2016 iPeM–integrated product engineering model in the context of product generation engineering. *Procedia CIRP* **50**, 100–105.

Allison, J. T., **Kokkolaras, M.** & **Papalambros, P. Y.** 2009 Optimal partitioning and coordination decisions in decomposition-based design optimization.

Ambrosio, J., **Darr, T.** & **Birmingham, W.** 1996 Hierarchical concurrent engineering in a multiagent framework, concurrent engineering. *Research and Applications* **4**(1), 47–57.

Belhe, U. & **Kusiak, A.** 1993 Performance analysis of design process using timed petri nets. *Concurrent Engineering: Research and Applications* **1**(3), 147–152.

Berker, I. & **Brown, D. C.** 1996 Conflicts and negotiation in single function agent-based design systems. *Concurrent Engineering: Research and Applications* **4**(1), 17–33.

Blanchard, B. S. 2004 *System Engineering Management*, Vol. **2004**. John Wiley & Sons: Hoboken, NJ, USA.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Cagan, J., **Campbell, M. I.**, **Finger, S.** & **Tomiyama, T.** 2005 A framework for computational design synthesis: Model and applications. *Journal of Computing and Information Science in Engineering* **5**(3), 171–181.

Campbell, M., **Cagan, J.** & **Kotovsky, K.** 1999 A-design: An agent-based approach to conceptual design in a dynamic environment. *Research in Engineering Design* **11**, 172–192.

Campbell, M. I., **Cagan, J.** & **Kotovsky, K.** 2003 The A-design approach to managing automated design synthesis. *Research in Engineering Design* **14**(1), 12–14.

Cardinal, L. B., **Turner, S. F.**, **Fern, M. J.** & **Burton, R. M.** 2011 Organizing for product development across technological environments: Performance trade-offs and priorities. *Organization Science* **22**, 1000–1025.

Christian, A. D. 1995 *Simulation of Information Flow in Design*. Ph.D. Thesis (ME), MIT: Cambridge, MA.

Christian, A. D. & **Seering, W. P.** 1995 A model of information exchange in the design process. In *Proceedings of the ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference)*, American Society of Mechanical Engineers (ASME) pp. 323–328.

Clarkson, J. P. & **Hamilton, J. R.** 2000 'Signposting', a parameter-driven task-based model of the design process. *Research in Engineering Design* **12**, 18–38.

Clarkson, P. J. & **Eckert C. M.** 2005 *Design Process Improvement: A Review of Current Practice*. Springer: London, UK, ISBN 185233701X.

Collopy, A. 2019 *Coordination strategies and individual behavior in complex engineered systems design*. Doctoral dissertation.

Collopy, A. X., **Adar, E.** & **Papalambros, P. Y.** 2020 On the use of coordination strategies in complex engineered system design projects. *Design Science* **6**, e32.

Crowder, R. M., **Hughes, H.**, **Sim, Y. W.** & **Robinson, M. A.** 2009 An agent-based approach to modelling design teams. In *Proceedings of ICED 09, the 17th International Conference on Engineering Design, Stanford, CA, 2009*. American Society of Mechanical Engineers (ASME)

Crowder, R. M., **Robinson, M. A.**, **Hughes, H. P. N.** & **Sim, Y. W.** 2012 The development of an agent-based modeling framework for simulating engineering team work. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* **42**(6), 1425–1439.

Crowder, R. M., **Sim, Y. W.**, **Payne, T. R.**, **Robinson, M. A.**, **Jackson, H.** & **del Carmen Romero Ternero, M.** 2008 An approach to modeling integrated product teams. In *Proceedings of the ASME Design Engineering Technical Conference, Brooklyn, NY, Aug. 2008*. American Society of Mechanical Engineers (ASME)

Cutkosky, M. R., **Engelmore, R. S.**, **Fikes, R. E.**, **Genesereth, M. R.**, **Gruber, T. R.**, **Mark, W. S.**, **Tenebaum, J. M.** & **Weber, J. C.** 1993 PACT: An experiment in integrating concurrent engineering systems. *Computer* **26**(1), 28–37.

Danesh, M. R. & **Jin, Y.** 2001 An agent-based decision network for concurrent engineering design. *Concurrent Engineering: Research and Applications* **9**, 37–47.

Devendorf, E. & **Lewis, K.** 2011 The impact of process architecture on equilibrium stability in distributed design. *Journal of Mechanical Design* **133**, 101001.

Elwert, F. 2013 *Graphical Causal Models. Handbook of Causal Analysis for Social Research* (ed. S. Morgan). Springer: Dordrecht, Netherlands.

Fender, J., **Graff, L.**, **Harbrecht, H.** & **Zimmermann, M.** 2014 Identifying key parameters for design improvement in high-dimensional systems with uncertainty. *Journal of Mechanical Design* **136**(4), 041007.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Fernandes, J. V., **Henriques, E.**, **Silva, A.** & **Pimentel, C.** 2017 Modelling the dynamics of complex early design processes: An agent-based approach. *Design Science* **3**, e19.

Grogan, P. T. & **de Weck, O. L.** 2016 Collaboration and complexity: An experiment on the effect of multi-actor coupled design. *Research in Engineering Design* **27**, 221–235.

Gurnani, A. & **Lewis, K.** 2008 Collaborative, decentralized engineering Design at the Edge of rationality. *Journal of Mechanical Design*, **130**(12), 121101.1–121101.9.

Haskins, C. 2006 *Systems Engineering Handbook*. INCOSE. The International Council on Systems Engineering: CA.

Hassannezhad, M., **Cantamessa, M.** & **Montagna, F.** 2015 Actor-based signposting: A modeling tool to improve the socio-technical design processes. In *Proceedings of the International Conference on Engineering Design, ICED*, Vol. **3**, pp. 165–174. Design Society.

Hassannezhad, M., **Cantamessa, M.**, **Montagna, F.** & **John Clarkson, P.** 2019 Managing sociotechnical complexity in engineering design projects. *ASME Journal of Mechanical Design* **141**(8), 081101.

Hirschi, N. & **Frey, D.** 2002 Cognition and complexity: An experiment on the effect of coupling in parameter design. *Research in Engineering Design* **13**(3), 123–131.

Hulse, D., **Tumer, K.**, **Hoyle, C.** & **Tumer, I.** 2018 Modeling multidisciplinary design with multiagent learning. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **33**(1), 85–99.

Jin, Y. & **Levitt, R. E.** 1996 The virtual design team: A computational model of project organizations. *Computational & Mathematical Organization Theory* **2**, 171–195.

Jin, Y. & **Lu, S. C. Y.** 2004 Agent based negotiation for collaborative design decision making. *CIRP Annals - Manufacturing Technology* **53**(1), 121–124.

Kang, N., **Kokkolaras, M.**, **Papalambros, P. Y.**, **Yoo, S.**, **Na, W.**, **Park, J.** & **Featherman, D.** 2014 Optimal design of commercial vehicle systems using analytical target cascading. *Structural and Multidisciplinary Optimization* **50**, 1103–1114.

Kapurch, S. J. 2007 *NASA Systems Engineering Handbook*. National Aeronautics & Space Administration (NASA): Washington D.C.

Kim, H. M., **Michelena, N. F.**, **Papalambros, P. Y.** & **Jiang, T.** 2003a Target cascading in optimal system design. *Journal of Mechanical Design* **125**(3), 474–480.

Kim, H. M., **Rideout, D. G.**, **Papalambros, P. Y.** & **Stein, J. L.** 2003b Analytical target cascading in automotive vehicle design. *Journal of Mechanical Design* **125**(3), 481–490.

Klügl, F. 2008 A validation methodology for agent-based simulations. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC'08) (16–20). Fortaleza, Ceara, Brazil*, ACM Digital Library pp. 39–43.

Königs, S. & **Zimmermann, M.** 2016 Resolving conflicts of goals in complex design processes - application to the design of engine mount systems. In *7th International Munich Chassis Symposium 2016*. Springer.

Korus, J. D., **Ramos, P. G.**, **Schütz, C.**, **Zimmermann, M.** & **Müller, S.** 2018 Top-down development of controllers for highly automated driving using solution spaces. In *9th International Munich Chassis Symposium 2018*. Springer

Krischer, L. & **Zimmermann, M.** 2021 Decomposition and optimization of linear structures using meta models. *Structural and Multidisciplinary Optimization* **64**, 2393–2407.

Krishnamachari, R. S. & **Papalambros, P. Y.** 1997a Hierarchical decomposition synthesis in optimal systems design.

Krishnamachari, R. S. & **Papalambros, P. Y.** 1997b Optimal hierarchical decomposition synthesis using integer programming.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Kunz, J. C., **Christiansen, T. R.**, **Cohen, G. P.**, **Jin, Y.** & **Levitt, R. E.** 1998 The virtual design team. *Communications of the ACM* **41**(11), 84–91.

Kusiak, A. 1999 *Engineering Design: Products, Processes, and Systems*. Academic Press.

Lapp, S., **Jablokow, K.** & **McComb, C.** 2019a KABOOM: An agent-based model for simulating cognitive style in team problem solving. *Design Science* **5**, e13.

Lapp, S., **Jablokow, K.** & **McComb, C.** 2019b Collaborating with style: Using an agent-based model to simulate cognitive style diversity in problem solving teams. In *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers.

Le, Q. & **Panchal, J. H.** 2011 Modeling the effect of product architecture on mass-collaborative processes. *Journal of Computing and Information Science in Engineering* **11**(1), 011003.

Lévárdy, V. & **Browning, T. R.** 2009 An adaptive process model to support product development project management. *IEEE Transactions on Engineering Management* **56** (4), 600–620.

Levitt, R. E. 2012 The virtual design team: Designing project organizations as engineers design bridges. *Journal of Organization Design* **1**(2), 14–41.

Levitt, R. E., **Thomsen, J.**, **Christiansen, T. R.**, **Kunz, J. C.**, **Jin, Y.** & **Nass, C.** 1999 Simulating project work processes and organizations: Toward a micro-contingency theory of organizational design. *Management Science* **45**(11), 1479–1495.

Lewis, K. & **Mistree, F.** 1997 Modeling interactions in multidisciplinary design: A game theoretic approach. *AIAA Journal* **35**(8), 1387–1392.

Lindemann, U., **Maurer, M.** & **Braun, T.** 2009 *Structural Complexity Management: An Approach for the Field of Product Design*. Springer-Verlag: Berlin, Heidelberg.

Liu, H., **Tang, M. X.** & **Frazer, J. H.** 2004 Supporting dynamic management in a multi-agent collaborative design system. *Advances in Engineering Software* **35**(8–9), 493–502.

Loch, C., **Mihm, J.** & **Huchzermeier, A.** 2003 Concurrent engineering and design oscillations in complex engineering projects. *Concurrent Engineering: Research and Applications* **11**(3), 187–199.

Macal, C. M. & **North, M. J.** 2008 Agent-based modeling and simulation: ABMS examples. In *Proceedings of the 2008 Winter Simulation Conference*, 7–10 Dec. 2008, IEEE: Miami, FL, USA.

Macal, C. M. & **North, M. J.** 2010 Tutorial on agent-based modelling and simulation. *Journal of Simulation* **4**, 151–162.

Maier, A., **Oehmen, J.** & **Vermaas, P. E.** 2022 *Handbook of Engineering Systems Design*. Springer.

Martinez-Miranda, J., **Aldea, A.**, **Bañares-Alcantara, R.** & **Alvarado, M.** 2006 TEAKS: simulation of human performance at work to support team configuration. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS06)*, ACM Digital Library: Hakodate, Japan, pp. 114–116.

Martínez-Miranda, J. & **Pavón, J.** 2012 Modeling the influence of trust on work team performance. *Simulation* **88**(4), 408–436.

McComb, C., **Cagan, J.** & **Kotovsky, K.** 2015 Lifting the veil: Drawing insights about design teams from a cognitively-inspired computational model. *Design Studies* **40**, 119–142.

McComb, C., **Cagan, J.** & **Kotovsky, K.** 2017 Optimizing design teams based on problem properties: Computational team simulations and an applied empirical test. *ASME Journal of Mechanical Design* **139**(4), 041101.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

McGuire, J., **Huokka, D.**, **Weber, J.**, **Tenenbaum, J.**, **Gruber, T.** & **Olsen, G.** 1993 SHADE: Technology for knowledge-based collaborative engineering. *Journal of Concurrent Engineering: Applications and Research* **1**(3), 137–146.

Meier, C., **Browning, T. R.**, **Yassine, A. A.** & **Walter, U.** 2015 The cost of speed: Work policies for crashing and overlapping in product development projects. *IEEE Transactions on Engineering Management* **62**(2), 237–255.

Meluso, J., **Austin-Breneman, J.** & **Shaw, L.** 2019 An agent-based model of miscommunication in complex system engineering organizations. *IEEE Systems Journal* **14**(3), 3463–3474.

Michalek, J. J. & **Papalambros, P. Y.** 2004 An efficient weighting update method to achieve acceptable consistency deviation in analytical target cascading. In *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, September 28–October 2, 2004*. American Society of Mechanical Engineers (ASME)

Michelena N., **Park, H.** & **Papalambros, P. Y.** 2003 Convergence properties of analytical target cascading. *AIAA Journal* **41**(5), 897–905.

Mihm, J., **Loch, C.** & **Huchzermeier, A.** 2003 Problem-solving oscillations in complex engineering projects. *Management Science* **49**, 733–750.

Mihm, J., **Loch, C. H.**, **Wilkinson, D.** & **Huberman, B. A.** 2010 Hierarchical structure and search in complex organizations. *Management Science* **56**(5), 831–848.

Nahm, Y.-E. & **Ishikawa, H.** 2004 Integrated product and process modeling for collaborative design environment. *Concurrent Engineering: Research and Applications* **12**, 5–24.

Olson, J., **Cagan, J.** & **Kotovsky, K.** 2009 Unlocking organizational potential: A computational platform for investigating structural interdependence in design. *Journal of Mechanical Design* **131**(031001), 1–13.

Pahl, G., **Beitz, W.**, **Feldhusen, J.**, **Grote, K. H.** 2007 *Engineering Design: A Systematic Approach*. 3rd edition. Springer-Verlag London Limited.

Panchal, J. H. 2009 Agent-based modeling of mass-collaborative product development processes. *Journal of Computing and Information Science in Engineering* **9**(3), 031007.

Panchal, J. H., **Fernandez, M. G.**, **Allen, J. K.**, **Paredis, C. J. J.** & **Mistree, F.** 2005 An interval-based focalization method for decision-making in decentralized, multifunctional design. In *ASME IDETC/CIE Advances in Design Automation Conference*, American Society of Mechanical Engineers (ASME): Long Beach, CA. Paper Number: DETC2005–85322.

Panchal, J. H., **Gero Fernández, M.**, **Paredis, C. J.**, **Allen, J. K.** & **Mistree, F.** 2007 An interval-based constraint satisfaction (IBCS) method for decentralized, collaborative multifunctional design. *Concurrent Engineering* **15**, 309–323.

Papalambros, P. Y. & **Wilde, J.** 2017 *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press.

Peterson, J. L. 1981 *Petri Net Theory and the Modeling of Systems*. Prentice Hall.

Raina, A., **Cagan, J.** & **McComb, C.** 2019 Transferring design strategies from human to computer and across design problems. *Journal of Mechanical Design* **141**(11), 114501.

Rebentisch, E., **Conforto, E. C.**, **Schuh, G.**, **Riesener, M.**, **Kantelberg, J.**, **Amaral, D. C.** & **Januszek, S.** 2018 Agility factors and their impact on product development performance. In *International Design Conference 2018*, The Design Society pp. 893–904.

Rivkin, J. W. & **Siggelkow, N.** 2003 Balancing search and stability: Interdependencies among elements of organizational design. *Management Science* **49**(3), 290–311.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

Rötzer, S., **Berger, V.** & **Zimmermann, M.** 2022b Cost optimization of product families using solution spaces: Application to early-stage electric vehicle design. *Proceedings of the Design Society* **2**, 583–592.

Rötzer, S., **Schweigert-Recksiek, S.**, **Thoma, D.** & **Zimmermann, M.** 2022a Attribute dependency graphs: Modelling cause and effect in systems design. *Design Science* **8**, e27.

Rötzer, S., **Thoma, D.** & **Zimmermann, M.** 2020 Cost optimization of product families using solution spaces. In *Proceedings of the Design Society: DESIGN Conference*, pp. 1087–1094. Design Society.

Safarkhani, S., **Bilionis, I.** & **Panchal, J. H.** 2020 Toward a theory of systems engineering processes: A principal–agent model of a one-shot, shallow process. *IEEE Systems Journal* **14**(3), 3277–3288.

Sargent, R. G. 1992 Validation and verification of simulation models. In *Proceedings of the 1992 Winter Simulation Conference* (ed. J. J.Swain, D.Goldsman, R. C.Crain & J. R. Wilson) ACM Digital Library.

Simpson, J. A. & **Weiner, E. S. C.** 1989 *The Oxford English Dictionary* (2nd ed., Vol. **3**). New York: Oxford University Press.

Singh, H., **Cascini, G.**, **Casakin, H.** & **Singh, V.** 2019 A computational framework for exploring the socio-cognitive features of teams and their influence on design outcomes. In *Proceedings of the 22nd International Conference on Engineering Design (ICED19), Delft*. The Netherlands: Cambridge University Press.

Singh, H., **Cascini, G.** & **McComb, C.** 2021 Influencers in design teams: A computational framework to study their impact on idea generation. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **35**, 332–352.

Singh, H., **Cascini, G.** & **McComb, C.** 2022 Idea selection in design teams: A computational framework and insights in the presence of influencers. *Design Science* **8**, e23.

Singh, V., **Dong, A.** & **Gero, J. S.** 2013 Social learning in design teams: The importance of direct and indirect communications. *AI EDAM* **27**, 167–182.

Smith, R. P. & **Eppinger, S. D.** 1997a Identifying controlling features of engineering design iteration. *Management Science* **43**(4), 276–293.

Smith, R. P. & **Eppinger, S. D.** 1997b A predictive model of sequential iteration in engineering design. *Management Science* **43**(8), 1104–1120.

Sobek, D. K., **Ward, A. C.** & **Liker, J. K.** 1999 Toyota's principles of set-based concurrent engineering. *Sloan Management Review* **40**(2), 67–83.

Soria, N. F., **Colby, M. K.**, **Tumer, I. Y.**, **Hoyle** & 2017 Design of complex engineered systems using multi-agent coordination. *Journal of Computing and Information Science in Engineering* **18**(1), 011003.

Summers, J. D. & **Shah, J. J.** 2010 Mechanical engineering design complexity metrics: Size, coupling, and solvability. *Journal of Mechanical Design* **132**(2), 021004.

Sycara, K. P 1991 Cooperative negotiation in concurrent engineering design. In *Computer-Aided Cooperative Product Development*, Springer pp. 269–297.

Toche, B., **Pellerin, R.** & **Fortin, C.** 2020 Set-based design: A review and new directions. *Design Science* **6**, e18.

Toye, G., **Cutkosky, M. R.**, **Leifer, L. J.**, **Tenenbaum, J. M.** & **Glicksman, J.** 1994 SHARE: A methodology and environment for collaborative product development. *International Journal of Intelligent and Cooperative Information Systems* **3**, 129–153.

Ulrich, K. T. & **Eppinger, S. D.** 2016 *Product Design and Development* (6th ed.). New York: McGraw Hill/Irwin.

Van Der Aalst, W. & **Van Hee, K. M.** 2004 *Workflow Management: Models, Methods, and Systems*. MIT Press: Cambridge, MA.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press

VDI 2206 2004 *Entwicklungsmethodik für mechatronische Systeme – Design methodology for mechatronic systems*. Beuth: Berlin.

Vermillion, S. D. & **Malak, R. J.** 2020 An investigation on requirement and objective allocation strategies using a principal–agent model. *Systems Engineering* **23**(1), 100–117.

Wagner, T. C. 1993 *A General Decomposition Methodology for Optimal System Design*. University of Michigan.

Wang, W., **Li, Y.**, **Li, H.** & **Liu, C.** 2012 An agent-based collaborative design framework for feature-based design of aircraft structural parts. *International Journal of Computer Integrated Manufacturing* **25**(10), 888–900.

Weilkins, T. 2007 *Systems Engineering with SysML/UML: Modeling, Analysis, Design*. Morgan Kaufmann OMG Press.

Wöhr, F., **Königs, S.**, **Ring, P.** & **Zimmermann, M.** 2020b A role-activity-product model to simulate distributed design processes. In *Proceedings of the 22nd International DSM Conference (DSM 2020)*. MIT: Cambridge, Massachusetts.

Wöhr, F., **Stanglmeier, M.**, **Königs, S.** & **Zimmermann, M.** 2020a Simulation of gradient-based individual design behaviour in distributed development processes. In *Proceedings of the Design Society: DESIGN Conference*, Cambridge University Press Vol. **1**, pp. 1579–1588.

Wöhr, F., **Uhri, E.**, **Königs, S.**, **Trauer, J.**, **Stanglmeier, M.** & **Zimmermann, M.** 2023 Coordination and complexity: An experiment on the effect of integration and verification in distributed design processes. *Design Science* **9**, e1.

Wynn, D. C. & **Eckert, C. M.** 2017 Perspectives on iteration in design and development. *Research in Engineering Design* **28**(2), 1–32.

Yassine, A., **Joglekar, N.**, **Braha, D.**, **Eppinger, S.** & **Whitney, D.** 2003 Information hiding in product development: The design churn effect. *Research in Engineering Design* **14**(3), 145–161.

Zare, A., **Michels, K.**, **Rath-Maia, L.** & **Zimmermann, M.** 2017 On the design of actuators and control systems in early development stages. In *8th International Munich Chassis Symposium, Proceedings* (ed. P. E. Pfeffer), Springer pp. 337–352.

Zhang, X. & **Thomson, V.** 2019 Modelling the development of complex products using a knowledge perspective. *Research in Engineering Design* **30**, 203–226.

Zimmermann, M., **Königs, S.**, **Niemeyer, C.**, **Fender, J.**, **Zeherbauer, C.**, **Vitale, R.** & **Wahle, M.** 2017 On the design of large systems subject to uncertainty. *Journal of Engineering Design* **28**(4), 233–254.

Zimmermann, M. & **von Hoessle, J. E.** 2013 Computing solution spaces for robust design. *International Journal for Numerical Methods in Engineering* **94**(3), 290–307.

https://doi.org/10.1017/dsj.2024.37 Published online by Cambridge University Press