Abstract

Determining a suitable level of description, or granularity, for a product or process model is not straightforward, especially since granularity can manifest in multiple ways, but it is important to capture important elements in the model without building models that are too large to understand. This article investigates the implications of model granularity choices by simulating the design process of a diesel engine on different levels of detail, comparing the results and exploring ways to account for the differences. It uses two Design Structure Matrix (DSM) models for change prediction in a diesel engine at different levels of granularity to run simulations of the design process. Changes are a major source of rework and lead to frequent rescheduling of design tasks. The incremental nature of product development as well as design changes and their propagation complicate design process planning further. Process simulation may provide support in such contexts when it is based on an appropriate description of the product. The paper shows that while coarse models can give an indication of likely process behaviour, they miss potentially significant iteration loops.

Experimental investigation of the implications of model granularity for design process simulation

1 Introduction

Contemporary product development is almost unthinkable in the absence of the insights drawn from modelling and simulation. In any modelling activity, there is usually a choice in the level of detail at which the model is constructed. This choice can have a significant impact on the effort required to build models. Building detailed models can require additional effort, as does the creation of more abstract models than easily afforded by the available data. This paper discusses the issue of granularity and uses the simulation of process models at different levels of detail to highlight that model granularity can have a significant effect on the outcome of simulation and needs to be considered both models are built and inferences are draw from them.

Modern engineering is largely carried out through the interaction with models. For instance, drawings, physical or CAD models are used to generate designs, and finite element analysis or computational fluid dynamics models can help evaluate the design. While these product models assist designers in creating innovative and functional products, other types of product models target the considerable interconnectedness of emerging products by capturing their underlying structure (e.g. Browning, 2001). With a similar aim, process models may capture the structural properties of design processes, such as information dependencies, to offer support for planning decisions, which are crucial for a timely delivery (Browning, 2002). However, process modelling and planning is also inherently difficult due to the uncertainty and complexity exhibited by design projects; and the relationship between the models and the processes they bring into being as epistemologically unclear (Eckert and Stacey, 2010).

The main motivation of process modelling is to gain an overview of the process, understand its behaviour and manage the process (Eckert and Clarkson, 2010). Building process models can involve considerable effort, as few engineering companies follow standard processes in their design activities close enough to use pre-existing models. Somebody has to gather data and development process models from typically heterogeneous sets of data and estimations based on past experience. The available data is often at different levels of detail and the modellers have to make a decision about what to include and at which level of detail the models are generated (Gericke et al., 2016). This process is difficult to automate. Data mining techniques allow to depict certain processes at maximum detail by mapping events that have occurred, however it present these techniques are largely applied to manufacturing or workflow processes and the date involve considerable effort to clean up (Suriadi et al. 2017). For the time being the practical challenge of choosing a meaningful and comprehensible

representation remains – in particular in design processes where many relevant steps do not leave digital traces. This leaves designers and researchers with little alternative to modelling design process in interaction with human experts. However, people often have little awareness that the level of detail of a model affects the actions the model affords and therefore the results that can be obtained with it (Maier, 2017). In particular, the level of detail affects the structure of the model in terms of the connectivity that is visible. The results of the paper will add to the model literature of people who build models and who use the results of models. The effects of simulations could either be the properties of the process or the properties of the models. In process model the structure reflects and affects the dynamic behaviour of the process as pointed out by Braha and Bar-Yam (2007).

To highlight the implications of deliberately choosing a specific level of detail, this paper demonstrates the differences between simulation results of the development of a new version of a diesel engine, based on a more or less detailed model. The models are change propagation models to be used in conjunction with the Change Propagation Method (CPM) developed at the Cambridge EDC (Clarkson et al, 2004). The models is a Design Structure Matrix (DSM) (see Browning, 2001) with added probability and impact of change propagating. The more detailed model has been built by the second author and her student with experts from the company (see Jarratt et al, 2004). The aim was to build a model that was both rich and easily comprehensible by looking at it on screen or a single piece of paper. The modelling process become with identifying a suitable level of detail in the product breakdown. This involved considerable negotiation with experts, as they wished to add small components that had caused problem in the past, such as lifting eyes, while the researcher wanted to keep a relatively even level of granularity. The model was built after consulting four engineers in collaboration with one of them. Later it was verified with the team by populated the probabilities. The preparatory process involved approximately 10 hours of expert interaction in addition to studying the structure of diesel engines in detail. The probability capturing exercise took between 4 and 7 engineers about 4 hours. As the example illustrates, selecting the right level of detail is important. Revisiting the same engineers to obtain probabilities at different levels of detail would have been very time consuming for the researchers and confusing for the engineers.

By nature, models are abstract representations of their target system, or the part of reality they aim to capture, created for a specific purpose (Frigg 2003). Depending on how and to what extent the target system is abstracted, the emerging model comprises a certain level of detail or *granularity*. The term granularity is used here to describe the level of detail in the description of various aspects of the target system (see Maier et al. 2016 for a review of definitions of granularity from a number of perspectives and disciplines).

The aim to this paper is to show through simulation the effect the choice of a particular level of detail has on the results; and thereby raise the awareness of practitioners and researchers

of the importance of selecting a suitable level of detail. Including all individual tasks (or parts) as separate entities would be as infeasible as modelling the entire process (or product) as one single entity. Between those extreme cases there is a range of options, which, depending on the modelling approach and the objective of the project, may lead to different models and consequently different results, which can impact related decisions. In process models, inaccurate results may lead to unfortunate planning decisions. This article thus investigates the implications of granularity choices by simulating the design process of the same artefact on different levels of detail, comparing the results and testing ways to account for the differences. As it is rarely possible to obtain models of the same product or process at different levels of detail in practice, therefore a coarser grained instance of the product model has been created using an aggregation algorithm proposed by Ariyo et al. (2007). Experimentation shows how more abstract models lead to simulation results with reduced variance, underestimating both delays due to iterations and impacts of policy interventions aimed at supporting task prioritisation decisions throughout the development process.

Complex systems, as often encountered in engineering design, are characterised as a multi-level hierarchy (e.g. Simon 1962; Alexander 1964; Sarkar et al. 2014), the choice of which level(s) to represent in a model is inherently difficult yet also of particular importance. Braha and Maimon (1998) point out the level of abstraction must be proportional to the information contained. The choice of model granularity cannot only affect the cost and effort to develop and maintain models (Pidd 1999; Robinson et al. 2004), but also the results of analyses based on these models (Chiriac et al. 2011; Suh et al. 2015). These choices may be particularly impactful in engineering design due to the influence such models have on their target system (see Eckert and Hillerbrand, 2018, for a discussion of the relationship between models to their target systems). For instance, product model granularity can influence modularisation of system architectures (Chiriac et al. 2011) or sequencing of integration tasks (Eppinger et al. 2014). Process models must account for the multi-disciplinary, interdependent, parallel and iterative nature of product development (Browning et al. 2006) and can exhibit considerable uncertainty (Wynn et al. 2011).

Within engineering design there is a distinction between product and process modelling, which are often handled separately but can also be integrated (see e.g. Eckert et al. 2015). In either case, granularity choices in one domain can influence choices in the other; equally, models within one domain can influence each other. For example, since change propagation in an artefact is an important factor in sequencing and executing design tasks, the granularity of a product model can influence design process simulation and consequently task prioritisation decisions (Maier et al. 2014; Maier et al. 2015). More recent studies have demonstrated the impact model granularity can have on analysis (Chiriac et al. 2011; Suh et al. 2015; Samy et al. 2015; AlGeddawy and ElMaraghy 2015). This highlights not only the practical importance of the topic, but also emphasises the need for accessible support that

captures and synthesises the various perspectives regarding model granularity and shows their pertinence for engineering design.

Despite the apparent importance of model granularity in the field of engineering design, relatively few research contributions address the topic directly. In practice, it is often assumed that some appropriate level will be determined without considering the sensitivity of the results (Chiriac et al. 2011). While some contributions emphasise the impact of model granularity on analysis (of product models), a systematic investigation of this topic within the area of process simulation is lacking. In order to better understand the implications of granularity for this important type of analysis in engineering design, this article presents experiments, investigating the impact of model granularity on design process simulation.

The experimentation in this article builds on a DSM-based design process simulation model (Maier et al. 2014) and enhances it by introducing different levels of model granularity. The model dynamically generates and simulates an array of potential activity sequences based on a product DSM, accounting for the combined effects of change propagation, design iteration, and rework (see Section 3 for a more detailed description). For the purpose of this experimentation, a detailed CPM model of a diesel engine, comprising 41 components, is aggregated to a more coarse-grained instance, comprising 10 systems (Section 4). Based on these two models, the design process of the diesel engine is simulated on two different levels of granularity to compare the results. Ways of accounting for the resulting differences by adjusting the simulation model are discussed, explored and their results compared (Section 5). Based on this illustrative example, the implications of varying the input model granularity for the design process simulation are reviewed. The impact of model granularity on analysis results and the emerging ramifications for making related decisions are discussed in more general terms (Section 6), before conclusions are drawn (Section 7).

2 Background

While it is generally accepted that the granularity of a model refers to its level of detail in some way, the term granularity and other related concepts are used in different ways across research communities. This section presents a characterisation framework for model granularity and provides a brief overview of relevant design process models to establish the background for the simulation model used and the experiments conducted in this study.

2.1 Characterising model granularity

The level of detail, or granularity, of a model is a topic that is relevant to range of disciplines relying on the use of models. As Maier et al. (2016; 2017) discuss, the terminology used to describe granularity and related concepts (e.g. abstraction, complexity, resolution), activities

that may influence granularity (e.g. aggregation, clustering, decomposition) and model characteristics resulting from granularity (e.g. accuracy, fidelity, precision) differs across these disciplines.

As the goal of this research is to support researchers in practitioners in building models at suitable levels of granularity the authors synthesise a framework outlining the main manifestations of model granularity. Figure 1 illustrates the resulting categories, principally distinguishing between *structural* and *information* granularity, see Maier et al. (2017) for details.

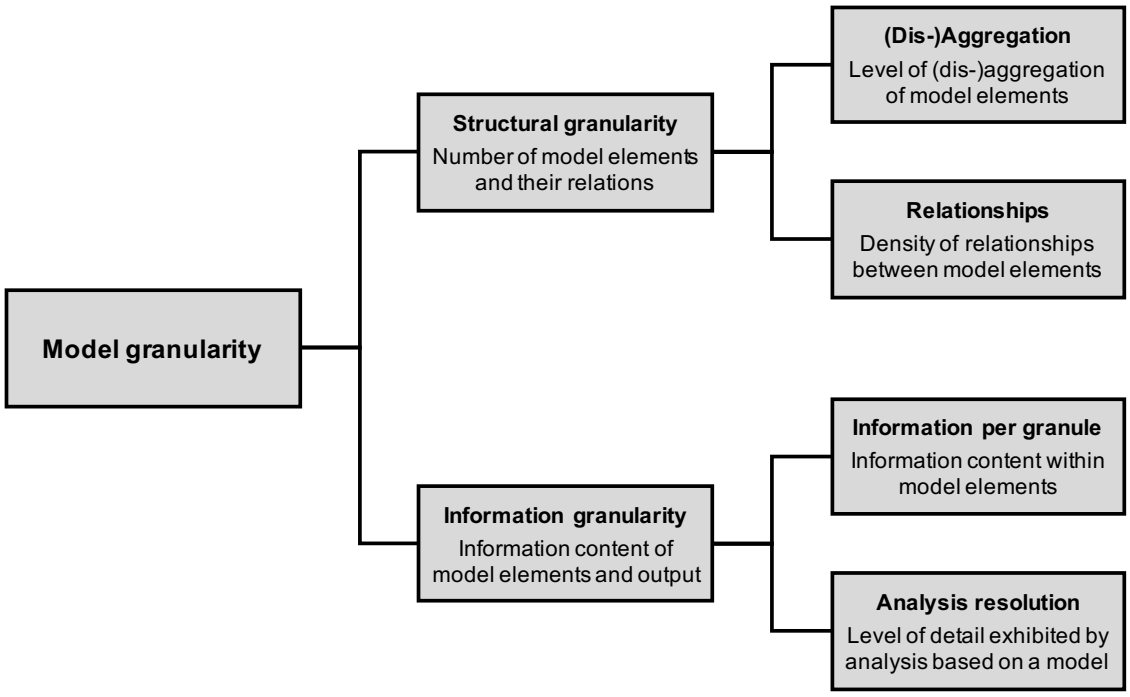

Figure 1: The main manifestations of model granularity from Maier et al. 2017

The structural dimension is derived from the definition of structures as a set of objects and a set of relations between them, as advocated by the structuralist view of models in philosophy of science (e.g. Frigg 2009). This definition does not make assumptions about the nature of these objects and their relations. Size and quantity of these objects as well as the number and nature of their relations depend on the level of abstraction of the model. The *structural granularity* of a model thus describes the level of decomposition of model elements and the density of relations between these. A *fine* granularity indicates many little elements, whereas *coarse* granularity indicates fewer, bigger elements. Similarly, the degree of detail in the description of relations can be referred to.

However, this structural dimension does not indicate the nature and content of model elements and their relations. A certain amount of information is required to describe individual elements, depending on the possible variety of their content (Shannon 1948).

Additionally, certain aspects of simulation models, such as dynamic properties or the statistical characteristics of their response surface cannot be fully captured by the structural dimension of granularity. *Information granularity* therefore refers to the intra-granule information content and the resolution of model-based analysis. Model elements with a higher information content have a finer information granularity. Analysis resolution generally indicates the amount of detail exhibited by model-based analysis based and may refer to the temporal resolution of dynamic simulation or the degree of discretisation of a process model.

Structural and information granularity are not necessarily independent of each other, as changes to one type of granularity may also affect the other. However, the distinction is useful nonetheless because it enables capturing a wider spectrum of modelling approaches while accounting for their particular properties. Maier et al. (2017) also indicate that structural and information granularity may be traded off (see Section 5).

The authors conceptualise granularity from the perspective of the information contained in the models. Others have proposed granularity metrics (e.g. Yao 2003) to quantify partitions of sets, which has also been used to quantify the granularity of process models (Holschke et al. 2009).

By contrast Braha and Maimon (1998) look at the artefact being designed and conceptualise abstraction in the context of a mathematical theory of design as a measure of the information context of a design representation, where a good design is seen as one without unnecessary complexity and a good representation of a design as one without unnecessary information. They follow Shannon and Wiener in seeing information as the reduction of uncertainty. They also note that "that the nature and number of abstraction levels are entirely dependent on the designer and the type of system being designed". They define the information associated with a design through the number of parameters required to describe a design, from which they can derive an entropy factor H. Abstract is the ratio between this absolute measure and the measure contained in a specific representation. While this definition is theoretically insightful, it is practically challenging to derive these factors for a complex existing product, like the diesel engine in the case, as it would require a complete detailed model of the final product.

2.2 Design process modelling and simulation

Design process modelling and simulation supports design engineers and managers in coping with uncertainty and complexity (Smith and Morrow 1999; Browning et al. 2006). The purposes of these models range from support for pricing and resource allocation to task prioritisation decisions and task planning. Process models are also repurposed throughout the design process. Models that start as a process plan and are used later to control and monitor the process. Finally, models are often the only record of the process, even though the

actual process might have been quite different to that modelled at the beginning (Eckert and Clarkson 2010).

Activity network models consider the order of activity execution, which is governed by information required and generated by each activity (Wynn 2007). Examples of activity network models include applications of Petri Nets (McMahon and Xianyi 1996), Signposting models and extensions (Clarkson and Hamilton 2000; Wynn et al. 2006) and approaches based on complex networks (Braha and Bar-Yam 2007). In activity-based models a range of valid decompositions may be derived and activities may be specified at varying levels of detail depending on the respective model context and purpose (Wynn 2007). This article focuses on an established and versatile technique of modelling the structural properties of processes and products: matrix-based models, in particular DSMs and related approaches which has also been employed outside of the process domain (e.g. Steward 1981; Eppinger et al. 1994). DSMs and extensions have been widely employed to model various aspects of product development (see Browning 2016 for a recent survey). Numerous articles apply DSM-based simulation, for instance to understand the impact of interdependencies and rework on the schedule and risk of design processes (e.g Smith and Eppinger 1997a; Browning and Eppinger 2002; Yassine 2007). Karniel and Reich (2009) note the ambiguities in transforming a DSM-based plan to a logically correct process model in the case of iterations. In their review of DSM-based process simulation models, they conclude that approaches differ in their simulation method, treatment of concurrency and underlying DSM type.

Simulation methods for DSM-based models can be distinguished in three main categories: deterministic, Markov chain and Monte-Carlo (Karniel and Reich 2009). *Deterministic* models assume that the DSM structure fully defines process progress. For example, Smith and Eppinger (1997b) consider the total work required to complete an intercoupled process using a deterministic model, in which the coupling strength between activities determines the amount of rework created on each time step. Yassine et al. (2003) extend this by reflecting developer's productivity levels as different completion rates per component. *Markov Chain* models describe a process as a memoryless progression between states according to transition probabilities. Smith and Eppinger (1997a) present a design process simulation using reward Markov chains based on DSM sequencing. *Monte-Carlo* models assume stochastic process behaviour according to model-specific logic. In their task-based discrete-event simulation model, Browning and Eppinger (2002) use Monte-Carlo methods to explore the impacts of activity sequence on cost and schedule risk. Cho and Eppinger (2005) additionally account for resource constraints and enhance representation of task concurrency and rework.

Treatment of concurrency can be distinguished in four categories, the first three of which take multipath approaches (Karniel and Reich 2009): (1) fully parallel execution of activities (e.g. Smith and Eppinger 1997b), (2) parallel with possible overlap (e.g Browning and Eppinger 2002; Yassine 2007), (3) serialised execution of activities (e.g Smith and Eppinger 1997a), and

(4) single-path approaches, assuming tasks are executed one-at-a-time (e.g Lévárdy and Browning 2009).

DSM types used for simulation modelling vary between three main categories (Karniel and Reich 2009): (1) binary DSMs (cannot directly be used as iteration probabilities), (2) probability DSMs using probabilities for simulation (Browning and Eppinger 2002), (3) numeric DSMs, for example indicating the number of iterations (Smith and Eppinger 1997b). Combinations of different types may be used, e.g. numeric and probability DSMs (Cho and Eppinger 2005).

3 Design process simulation model used for experimentation

Planning design processes is a difficult undertaking throughout product development projects. It involves a range of activities, from strategic planning decisions regarding system architecture selection to operational scheduling of design task sequences (Wynn 2007). In many cases, designers require a quick method to estimate development effort based on the current state of the design. Examples include planning iteration loops, selecting between system architecture alternatives or choosing among ways of implementing a change to respond to unexpected problems or new customer requests. The required design tasks differ in their nature. While some tasks like testing are largely independent of potential alternatives, others are contingent on the chosen solution or on the nature of necessary changes. Process simulation can play an important role in process analysis: to understand the dynamic behaviour of design processes or to understand the risk of process running over time. It can for example help to identify bottle necks in processes (Braha and Bar Yam 2007), which would need to be removed to make significant improvements. Simulation also generates a multitude of specific process plans, from which users can select a process that meets their constraints and preferred level of risk.

Taking the effects of changes on both product and process into account enables the designer to consider process effort when considering modifications to an existing product or making decisions about design task prioritisation. A small number of integrated product and process models exist (Eckert et al. 2017), but most of them do not incorporate product specific information for process simulation. To address this, a discrete-event simulation model was developed (Maier et al. 2014), which simulates potential activity sequences based on a product DSM. These activity sequences represent the maturity progression of components of the design. This research builds up on this model and substantially extends it by introducing different levels of model granularity. It brings together three important issues in design to study their combined effects:

- iteration carried out to progress the design (Smith and Eppinger 1997a);

- iteration, or rework, necessary to correct errors or address design changes (e.g. Wynn et al. 2007);
- change propagation due to structural interdependencies in the developed product (Clarkson et al. 2004).

The simulation assumes an incremental design problem where a new product is generated or an existing product is modified while maintaining a stable system architecture. The level of maturity associated with a component may be thought of as its degree of readiness to be employed in the final design. Changes are initiated randomly, propagate depending on the design structure and impact component maturity. This makes it possible to use a change prediction matrix (see Clarkson et al. 2004) to guide the simulation of design processes. The model is briefly described here, with reference to the framework presented in Section 2.1.

3.1 Overview of the simulation model

The model enables a design process simulation on the basis of product DSMs, representing n components and their interfaces in a design (Browning 2001). It synthesises the iterative progression between maturity level by randomly initiating change during task completion. The model uses Monte Carlo methods to simulate occurrence and propagation of changes and treats concurrency using a multipath approach, executing activities in parallel with potential for overlap. Figure 2 presents an overview of the simulation algorithm.

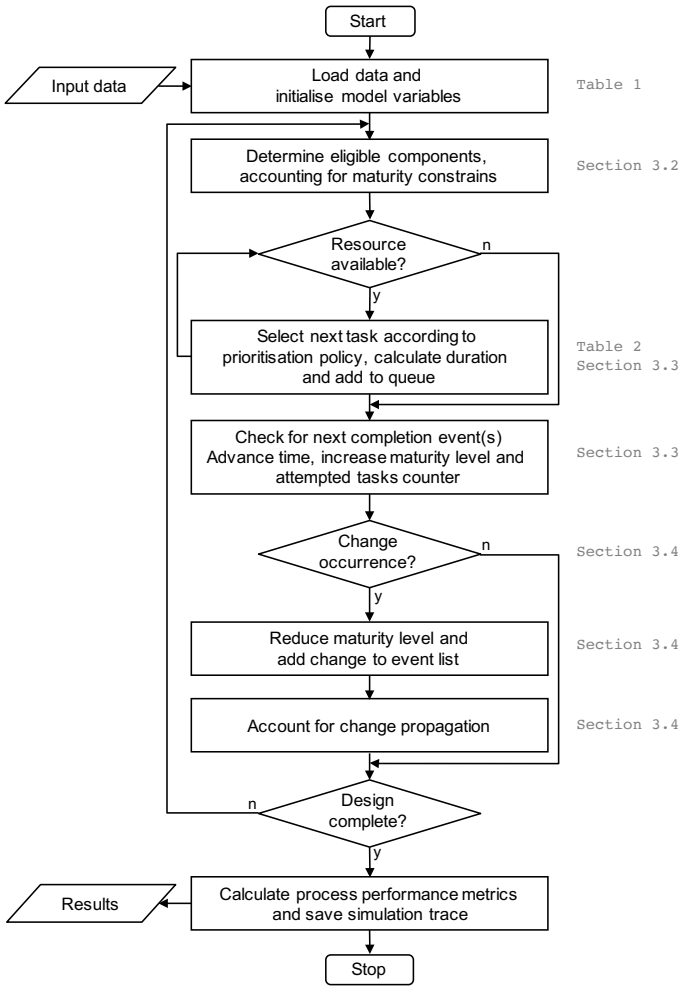

Figure 2: Overview of the algorithm for a single run of the Monte Carlo simulation

3.2 The DSMs

Numeric entries between 0 and 1 describe the impact and likelihood of change propagation associated with each dependency (Clarkson et al. 2004). If this level of information granularity is not available and a binary DSM is used as input, these values are assumed to be both 0.5. In addition to the information in the DSM, another input to the models is the amount of time it takes to develop each component in the absence of resource limitations, changes and rework. Maturity levels assigned to each component track the state of design progression, which changes during the simulation. Using m discrete maturity levels allows for $m-1$ transformations between them. Work on a component increases its maturity level, while a decrease may be triggered by changes, which are modelled as random events. The main cause for delays in the simulated design process is thus rework resulting from changes and their propagation to other connected parts of the design. The simulation ends when all components reach their maximum maturity level m.

The simulation algorithm can be expressed as a cycle of three steps (explained in more detail in Sections 3.3 - 3.5), which is repeated until the simulation is complete:

1. Identify task(s) to start: account for maturity and resource constraints and make prioritisation decisions.
2. Start task(s): calculate task durations including learning effects and update model state to start tasks.
3. Complete task(s): simulate change initiation and propagation upon task completion and update model state accordingly.

Table 1 presents an overview of model parameters and the symbols used to refer to them. Figure 2 summarises the simulation algorithm.

Table 1: Model parameters referred to in this article, including standard values (see Maier et al. 2014)

Parameter	Description	Definition	Std.
m	No. of maturity levels	$m \in \mathbb{N}$	5
Δm_{max}	Max. allowed maturity level difference	$\Delta m_{max} \in \mathbb{N} \mid \Delta m_{max} < m$	2
p_c	Probability of change initiation	$p_c \in \mathbb{R} \mid 0 \leq p_c \leq 1$	0.1
s_{max}	Max. no. of change propagation steps	$s_{max} \in \mathbb{Z}$	5
n	No. of components in the design	$n \in \mathbb{N}$	n/a
q	No. of resources	$m \in \mathbb{N} \mid q \leq n$	n/a

$\mathbb{Z}$ is the set of all integers. $\mathbb{N}$ is the set of all nonzero positive integers. $\mathbb{R}$ is the set of all rational numbers.

3.3 Identify tasks to start

Which tasks can be started at any given point in the development process is determined by each component's maturity level, availability of appropriate resources as well as the relative priorities of other components, requiring the same resource.

Accounting for maturity constraints. The sequence of design progress is constrained by the design structure and components' current maturity. The model assumes that interconnected components in the DSM cannot be designed independently and have progress together. A component can only be selected for further work if the maturity levels of all other components it is dependent on are at most one level lower than its own ($\Delta m_{max} = 2$). Thus, in the absence of rework, the maturity of any component would never be more than two levels higher than any of the components it is dependent on.

Accounting for resource constraints. Resources are required to execute design activities. In the model, resources are interpreted as individual designers or teams, which can work on progressing the respective component. The model allows for assigning specialised skills to resources, allowing them to work on specific components, or for assuming that every resource

has generic skills and can work on every component. An eligible component can only be progressed at the current time if at least one resource is both idle and capable of working on it. The number of resources is specified by parameter q.

Making priority decisions. If there are more eligible components than available resources to work on them, a priority decision has to be made. This decision is simulated using a function that evaluates the priority for each eligible task. The task with the highest priority is chosen to execute and the others must wait. 25 prioritisation rules are defined (partly adapted from Braha and Bar-Yam 2007) consistently prioritising the task with either the minimum or maximum value for the selected decision criterion (Table 2). The active sum of a DSM is calculated by summing up the respective entries (likelihood, impact or risk) across a column, which represents all outgoing connections of a component. The passive sum is calculated by summing up the entries across a row, representing the incoming entries for a component. For binary DSMs, policies 9–20 are identical to policies 5–8, and for symmetric DSMs, active and passive sums are similar. If two or more tasks have identical values, a random tiebreaker chooses between them.

Table 2: Decision policies and corresponding criteria of components to be prioritised

Decision criterion	**Prioritisation rule**	
None	0: Random selection	
Task duration	1: Shortest first	2: Longest first
Current maturity level	3: Lowest first	4: Highest first
Active sum in binary DSM	5: Lowest first	6: Highest first
Passive sum in binary DSM	7: Lowest first	8: Highest first
Active sum in risk-DSM	9: Lowest first	10: Highest first
Passive sum in risk-DSM	11: Lowest first	12: Highest first
Active sum in impact-DSM	13: Lowest first	14: Highest first
Passive sum in impact-DSM	15: Lowest first	16: Highest first
Active sum in likelihood-DSM	17: Lowest first	18: Highest first
Passive sum in likelihood-DSM	19: Lowest first	20: Highest first
Total attempts	21: Fewest first	22: Most first
Total amount of rework	23: Lowest first	24: Highest first

3.4 Start tasks

After the task(s) to be started by each idle resource are chosen as explained above, the durations are calculated and the model state is updated.

Calculating task durations accounting for learning effects. The estimated development durations for each component are an input to the model. These durations do not include possible delays due to resource limitations, changes and rework, as this is calculated during the simulation. If a maturity level is reached for the first time, a fixed and identical proportion (determined by the number of possible maturity transitions) of the total duration is required. Reworking an activity often takes less time than to attempt it for the first time (Browning and Eppinger 2002). The duration of a task accounting for a change, thus returning it to a previously reached maturity level, is subject to learning effects. The improvement curve is adapted from Cho and Eppinger (2005), assuming a linear improvement by a certain percentage (25%) on each consecutive attempt until a minimum fraction (25%) of the original duration is reached.

Updating model state to start tasks. For each resource, the task to work on is identified from the list of tasks to be started. Tasks are added to the event list, accounting for their duration. Resources are flagged as *occupied* and tasks as being *worked on*.

3.5 Complete tasks

After starting the chosen tasks, the simulation advances to the next task that has reached the end of its execution time and is due for completion. The state of the model is updated accordingly, and potential change initiation and propagation is considered. Once this has been processed, resources may have become available and new tasks eligible to be started, effectively returning the model to the first step (Section 3.3).

Updating model state to complete tasks. The completed task is removed from the event list, the simulation time is advanced, the respective component is flagged as not being worked on and the responsible resource as idle, the maturity level and total iteration counter for the component are increased. If multiple completions occur at the same time, they are processed similarly.

Change initiation at task completion. Changes are modelled as random events when activities are completed, which is thought to be a realistic assumption (Browning and Eppinger 2002; Yassine et al. 2001). This logic is used to represent changes both internal and external to the design process. A change in a component that is being worked on results in a task interruption and release of the respective resource. The probability p_c of change initiation upon task completion is constant. The model allows for two assumptions regarding the affected component in case a change occurs: affects component that has just been worked on

(standard assumption) or any component, selected randomly. Changes are taken into account by reducing the affected component's maturity level.

Change propagation at task completion. Initiated changes may propagate to other, interconnected components. The model assumes that this occurs within the same timestep as the initiated change. The logic of the CPM is used to simulate the knock-on effects of an initiated change. The entries in the likelihood DSM determine the probability of changes propagating from the initiating component to components that are dependent on it. A Monte Carlo approach simulates whether any of these propagations occur. Second- and higher-order propagations are simulated by repeating this process for affected components. When a change propagates from one component to another, the maturity level of the latter is reduced according to the respective entry in the impact DSM. The higher the degree of connectivity and the likelihoods of change propagation, the more extensive the resulting propagation tree is likely to be (Clarkson et al. 2004; Braha and Bar-Yam 2007). Change propagation is terminated when it exceeds s_{max} (standard: five) steps, given that changes would not (be allowed to) propagate infinitely in practice (Clarkson et al. 2004; Pasqual and de Weck 2011). Change propagation also terminates when the algorithm revisits a component that has already been subject to a change in the current timestep.

4 Change propagation on different levels of granularity

To generate a product DSM on different levels of granularity, which can be used as an input for the simulation model presented in Section 3, an approach from Ariyo et al. (2007) is adapted. Predicting changes on different levels of granularity is a challenging problem, given that the structural granularity of the underlying model determines system boundaries and propagation paths. Intra-system propagation may be obscured by more abstract models and the impacts of change propagation cannot easily be transferred between models on different levels of granularity. Top-down approaches generally require additional data collection and do not follow a standard procedure. The bottom-up approach taken here allows to aggregate an existing detailed change propagation model with the help of a multi-step algorithm. The algorithm as well as the original and aggregated model of a diesel engine, which is used in the experimentation (Section 5), are outlined in this section.

4.1 Aggregating change propagation models

Ariyo et al. (2007) introduce an algorithmic approach, which allows change prediction on different levels of granularity, as an extension to the CPM (Clarkson et al. 2004). This is realised through bottom-up aggregation, resulting in a coarser model instance. The approach

distinguishes between inter- and intra-system connectivity (Figure 3). This allows the calculation of the likelihoods of changes propagating from components-to-systems, system-to-components and systems-to-systems (Figure 5). This is a pragmatic choice of algorithm, since computing these independently is the basis for assessing change propagation likelihoods on various levels of granularity.

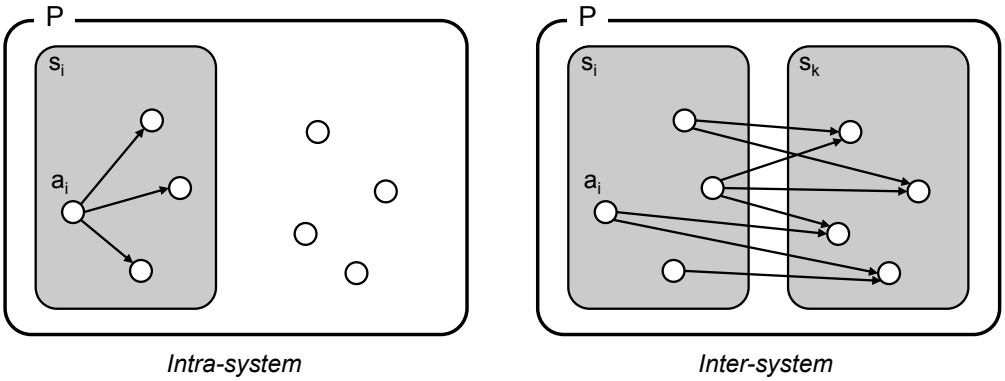

Figure 3: Intra- and inter-system connectivity (adapted from Ariyo et al. (2007))

Calculating likelihoods of change propagation on different granularity levels may reduce the development effort of hierarchical models and ensures consistent estimation of change likelihood across levels. However, the algorithm does not allow the estimation of change propagation impacts on different levels. The expectation is that a system-to-system change propagation has a higher impact than a component-to-component propagation but this is not accounted for. Measures of change impact may indicate the amount of necessary rework (Section 3.5). There is, however, no indication of the effect of changes on the design process. On a system level, intra-system change propagation is not represented. This could conceal parts of the change propagation behaviour and resulting rework, especially in highly modular products. Simulating the design process of the respective product, taking these effects into account, may provide a better understanding of change propagation behaviour (Section 5).

The first step of the algorithm introduced by Ariyo et al. (2007), is to compute the combined likelihoods of changes propagating between two components across a system boundary. As Figure 4 indicates, a change initiating in component a_i in system S_A may propagate via a 'boundary' component a_k, which is connected to component b_j in system S_B.

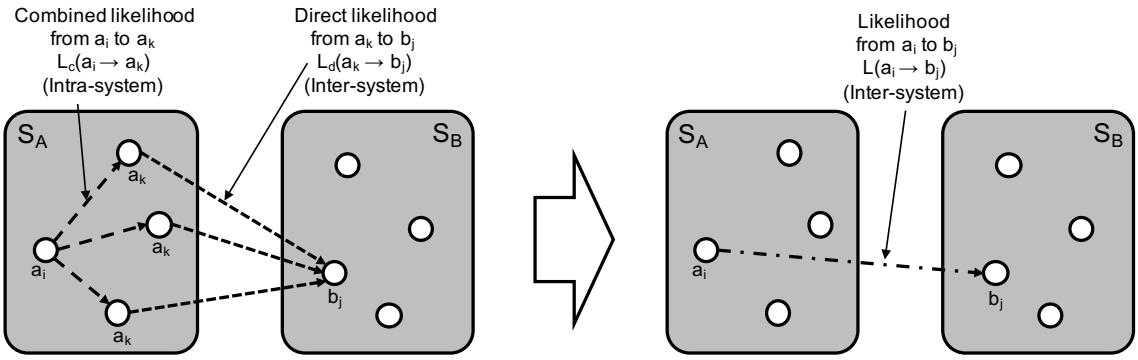

Figure 4: Estimating the component-to-component inter-system likelihood (adapted from Ariyo et al. (2007))

The combined intra-system component-to-boundary-component likelihoods $L_c(a_i \rightarrow a_k)$ are calculated with the CPM algorithm described by Clarkson et al. (2004). Multiplying this value with the direct likelihood of changes propagating from boundary components to components in system S_B, yields the likelihood of component a_i in system S_A affecting component b_j in system S_B through a specific boundary component a_k in system S_A:

$$L\left(a_i \rightarrow a_k \rightarrow b_j\right) = L_c(a_i \rightarrow a_k) \cdot L_d\left(a_k \rightarrow b_j\right) \quad (1)$$

Based on this, the likelihood of a change in a_i affecting b_j via all possible boundary components a_k can be determined:

$$L\left(a_i \rightarrow b_j\right) = 1 - \prod_{k=1}^{n}\left[1 - L\left(a_i \rightarrow a_k \rightarrow b_j\right)\right] \quad (2)$$

This inter-system component-to-component likelihood $L(a_i \rightarrow b_j)$ is then used to calculate component-to-system, system-to-component and system-to-system likelihoods. Figure 5 illustrates how these likelihoods are estimated.

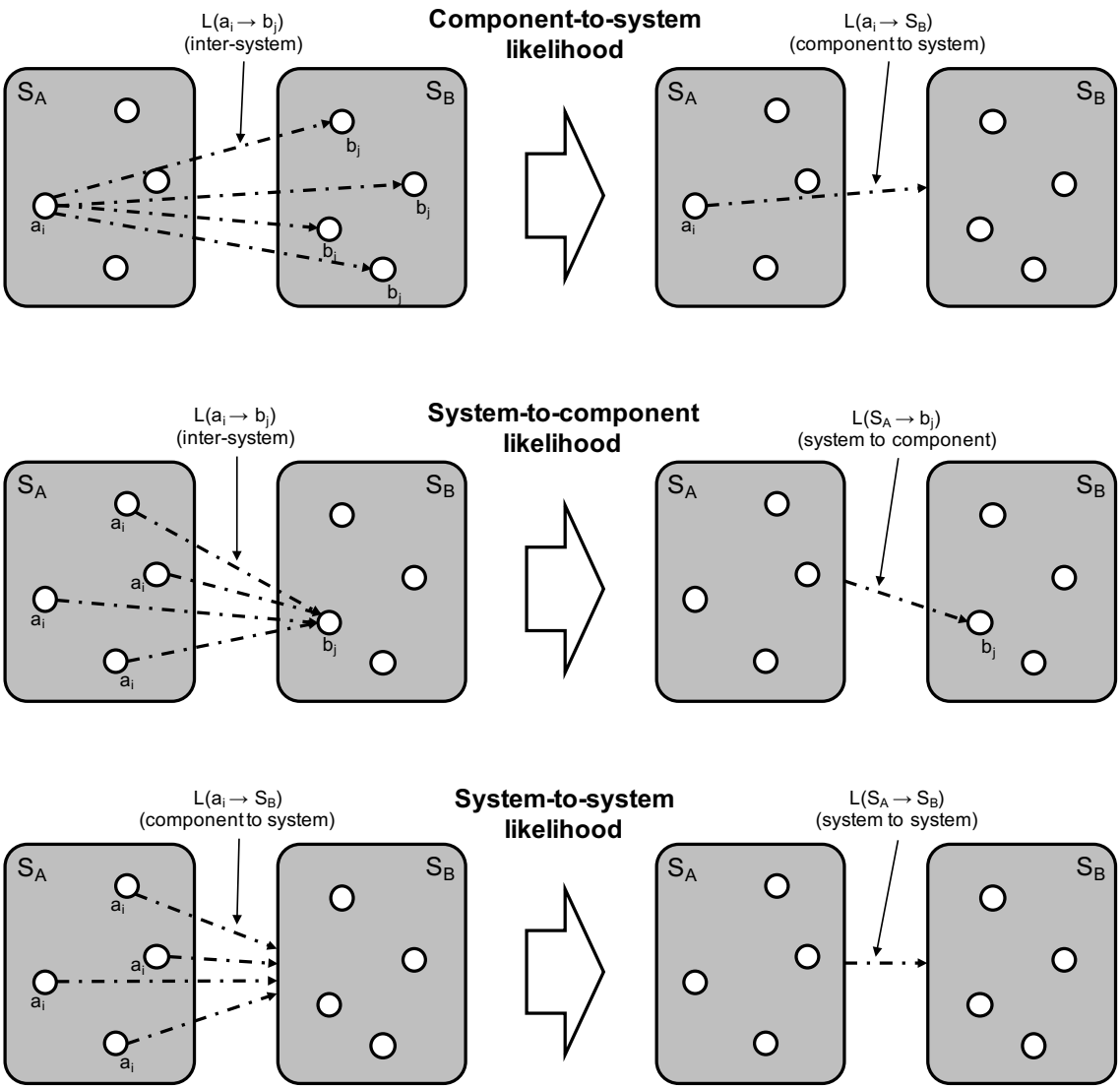

Figure 5: Estimating component-to-system, system-to-component and system-to-system likelihoods (adapted from Ariyo et al. (2007))

The likelihood of a change propagating from a component a_i to system S_B is calculated by computing a multiplication over $L(a_i \rightarrow b_j)$ obtained from Equation (2) for any component in S_B:

$$L(a_i \rightarrow S_B) = 1 - \prod_{j=1}^{n} \left[1 - L\left(a_i \rightarrow b_j\right)\right] \tag{3}$$

Estimating system-to-component likelihoods is conceptually more difficult because a change to a system does not necessarily affect all of its components. It is thus important to also consider the probability of a change initiating within a system. Ariyo et al. (2007) note that logical problems would result from aggregating likelihoods as in the two previous equations and suggest using numerical averages to obviate this:

$$L\left(S_A \rightarrow b_j\right) = \frac{1}{n}\sum_{i=1}^{n} L\left(a_i \rightarrow b_j\right) \tag{4}$$

Similarly, the system-to-system likelihood is the average of all component-to-system likelihoods, as calculated in Equation (3):

$$L(S_A \rightarrow S_B) = \frac{1}{n}\sum_{i=1}^{n} L(a_i \rightarrow S_B) \tag{5}$$

Based on these calculations, a system-level likelihood DSM can be computed, based on a component-level DSM (outlined in Section 4.2).

4.2 Used product model with two levels of structural granularity

For the purpose of this experimental investigation, the product model of a diesel engine is used, as described by Jarratt et al. (2004) and adopted by Ariyo et al. (2007). This is employed as an illustrative example with two levels of granularity to show the underlying principles. Other product models and hierarchical decompositions can be processed by the simulation model in a similar manner. Figure 6 shows the component-level DSM of the diesel engine, comprising 41 components. The cells of this matrix indicate the direct likelihoods of changes propagating between components (from column to row) and are shaded accordingly. The DSM also indicates which components belong to which system, highlighting the underlying hierarchical structure. The specific hierarchical decomposition depends on the context and purpose and may require some negotiation, as Ariyo et al. (2007) note. With the help of domain experts, they group the 41 components into ten systems, which is adapted in this work.

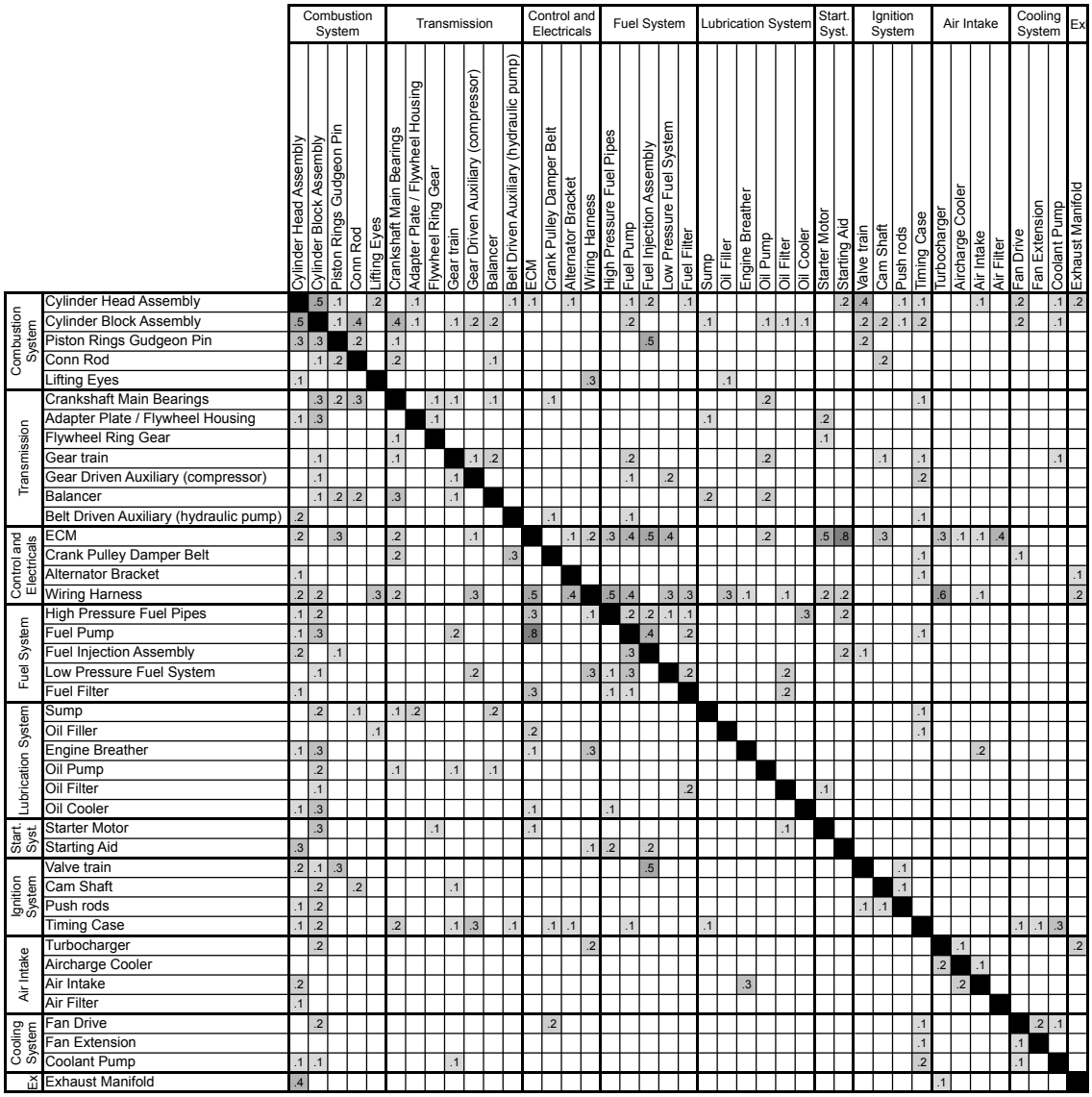

Figure 6: Component-level DSM of a diesel engine; numbers and shading indicate change propagation likelihoods between components

Based on the component-level DSM shown in Figure 6, numeric values for system-level change propagation likelihoods can be computed, using the algorithm outlined in Section 4.1. This leads to a coarser structural granularity of the model. The calculations were performed using a custom implementation of the algorithm in a MATLAB script. Running this algorithm yields a 10x10 system-level DSM of the diesel engine (Figure 7), indicating the likelihoods of change propagation between systems (rounded to two decimal places).

	Combustion System	Transmission	Control and Electricals	Fuel System	Lubrication System	Starter System	Ignition System	Air Intake	Cooling System	Exhaust System
Combustion System		.22	.19	.26	.08	.10	.39	.03	.23	.20
Transmission	.57		.05	.19	.13	.14	.13		.04	
Control and Electricals	.42	.18		.69	.12	.72	.13	.39	.04	.28
Fuel System	.40	.06	.45		.11	.18	.05			
Lubrication System	.46	.11	.25	.09		.05	.05	.06		
Starter System	.24	.02	.08	.12	.02					
Ignition System	.49	.12	.05	.16	.02				.19	
Air Intake	.20		.10		.05					.20
Cooling System	.17	.02	.05				.09			
Exhaust System	.17							.03		

Figure 7: System-level DSM of a diesel engine; numbers and shading indicate change propagation likelihoods between systems

5 Experimental design and results

The implications of different levels of model granularity for design process simulation are explored in a set of simulation experiments, using the two diesel engine models (Section 4.2) as inputs. This yields a perspective on the potential effect of model granularity on analysis results, with reference to a specific model. As shown by Maier et al. (2014), the structural granularity (size and density) as well as the information granularity (binary vs. impact and likelihood values) of the product model can have an impact on the analysis results.

The presented experiments have two purposes. Firstly, they aim to explore the effect of different structural granularities on results by using the model of a diesel engine on two different levels of granularity (Section 4.2). Secondly, four ways of accounting for these differences in structural granularity are analysed and further options are discussed. In addition to adapting model factors representing basic assumptions (Section 5.2 a & b), the information granularity of the simulation model is altered in the experimentation (Section 5.2 c). An overview of the conducted experiments is given in Table 3, indicating the set-up of the most important simulation parameters and highlighting how they are manipulated compared to a baseline simulation with the original, fine-grained model (see Section 5.2 for a more detailed explanation of the differences between the respective experiments). The number of resources is limited to one, even though the simulation allows for concurrency. While this simplifies conditions, it enables to study the impact of varying granularity without influence from concurrency effects (e.g. overlaps between tasks and change propagation across several teams working on parts of the overall design simultaneously). The results of all experiments

are based on 10,000 simulation runs with Monte-Carlo sampling and the maximum number of change propagation steps s_{max} is set to five. An overview of a typical distribution of simulation results for all experimental set-ups is provided in Figure 13 to facilitate comparison.

Table 3: Overview of the simulation experiments with shaded cells indicating manipulated parameters

Pre-set parameters	**Section 5.1**	**Section 5.1**	**Section 5.2 a)**	**Section 5.2 a)**	**Section 5.2 b)**	**Section 5.2 c)**
Input product model	Fine	Coarse	Coarse	Coarse	Coarse	Coarse
Priority policy (Table 2)	0 – 24; 0	0 – 24; 0	0	0 – 24; 0	0 – 24; 0	0 – 24; 0
Probability of initiating change (p_c)	0.1	0.1	0.1 - 0.32	0.3	0.1	0.1
Max. change propagation steps (s_{max})	5	5	5	5	5	5
Max. allowed maturity difference (Δm_{max})	2	2	2	2	2	8
Number of maturity transitions (m-1)	4	4	4	4	4	16

5.1 Simulation results with basic experimental set-up

To enable an experimental comparison of the two input models (Section 4.2), some assumptions have to be made. In particular, the expected task durations have to be adjusted. To account for this, the durations to develop individual components are added up to estimate how long it would take to develop the system they are part of. Given that the original diesel engine model does not indicate the necessary design effort for its components, it is assumed that their development requires equal amounts of time (in the absence of iterations and rework). Developing a system is consequently assumed to require this amount of time multiplied by the number of components it comprises. Figure 8 depicts policy performance relative to random task selection (Table 2) as well as a histogram of total project duration with policy 0 and 10,000 simulation runs with Monte-Carlo sampling.

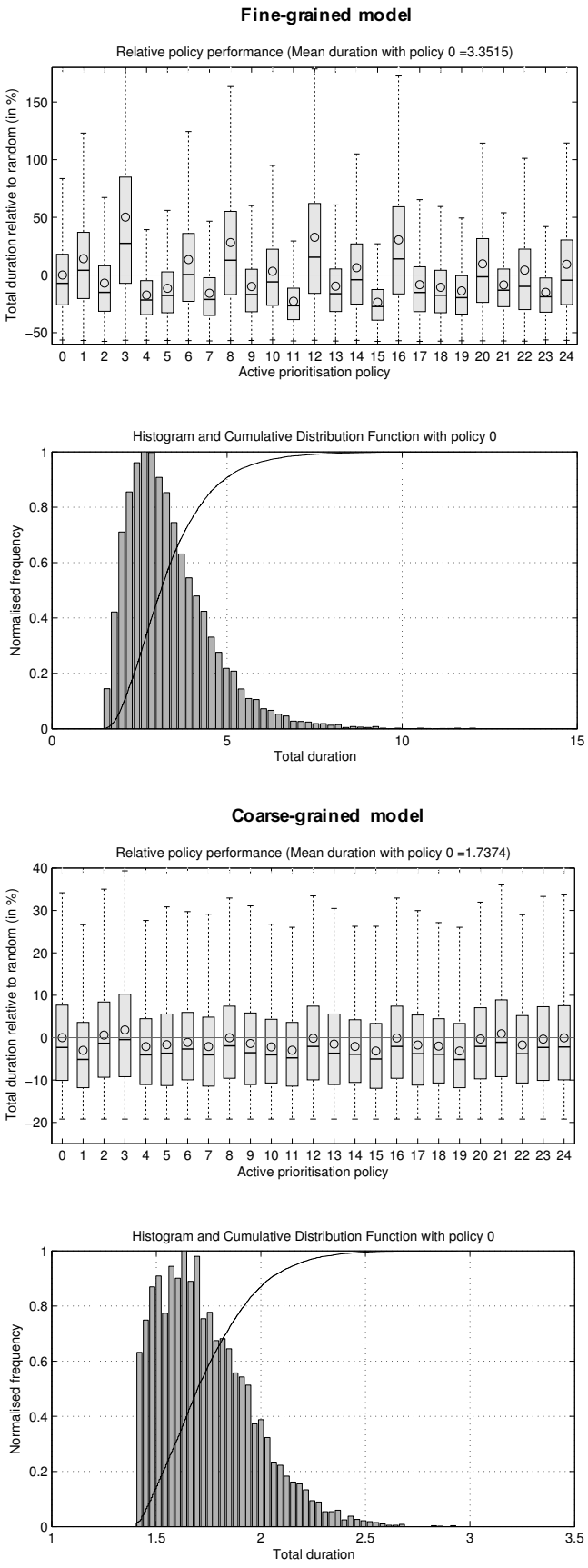

Figure 8: Simulation results with basic simulation parameters. Boxplot of relative policy performance and histogram for 10,000 simulation runs with policy 0 (random selection) and total duration in years

Figure 8 illustrates the differences between the original and aggregated model, in particular the smaller variation in relative policy performance and the decrease in both mean and variance of the total duration. Given that performance is measured in terms of project duration, these effects are not independent. Despite the less pronounced differences, policy performance remains qualitatively similar in the aggregated model. An exception are policies based on task durations, which are identical in the original model. The lower mean duration in the aggregated model can be explained by multiple factors. Firstly, assuming that the expected duration to develop a system equals the sum of durations of its components, simplifies actual conditions and conceals intra-system iteration, change propagation and rework (see also Section 4.1). Secondly, fewer changes are initiated in the aggregated model when assuming an identical likelihood p_c, given that this only occurs upon task completion. Thirdly, the number of total maturity progression steps ($m \cdot n$) is lower in the aggregated model, because the number of maturity levels m is identical for both models with different numbers of components n. Finally, change propagation impacts may be underestimated in the aggregated model. Due to a lack of both information and a validated mechanism to aggregate them from the detailed model, they are assumed to be uniform across the two models.

5.2 Simulation results with adjusted experimental set-up

After comparing simulation results for the two product models in the previous section, the question is how and to what extent the difference in structural input model granularity can be accounted for by adjusting model assumptions. As the previous section outlined, the main parameters to adjust the simulation model are:

a) Likelihood of change initiation p_c,

b) Aggregation of task durations on system level,

c) Number of maturity levels m per system vs. component and

d) Impacts of change propagation on a system level.

Essentially, these adjustments lead to a finer information granularity by incorporating information that can be inferred or estimated by comparing the structural granularity of the original and aggregated model. In particular, a), b) and d) increase the information content behind the respective model factors, while c) increases the simulation resolution. These modifications do not alter the structural granularity of the model.

a) Likelihood of change initiation p_c

Out of these parameters, varying the likelihood of change initiation is the only one that allows for a systematic sensitivity analysis by varying the respective parameter, without the necessity

to introduce additional assumptions. Figure 9 displays the results of a sensitivity analysis, carried out by varying p_c in a range from 0.1 (standard) to 0.32, with results of the original model on the left side for comparison (see also Figure 8). The resulting mean duration of the aggregated model with $p_c = 0.3$ corresponds to the original model with $p_c = 0.1$. However, the characteristics of the distribution are different. The results for the original model (and the aggregated model with $p_c = 0.3$) are: mean 3.35 (3.35), median 3.08 (3.17), lower quartile 2.47 (2.62) and upper quartile 3.93 (3.84). This indicates that the detailed model's results are more widely spread and positively skewed (longer tail of the distribution).

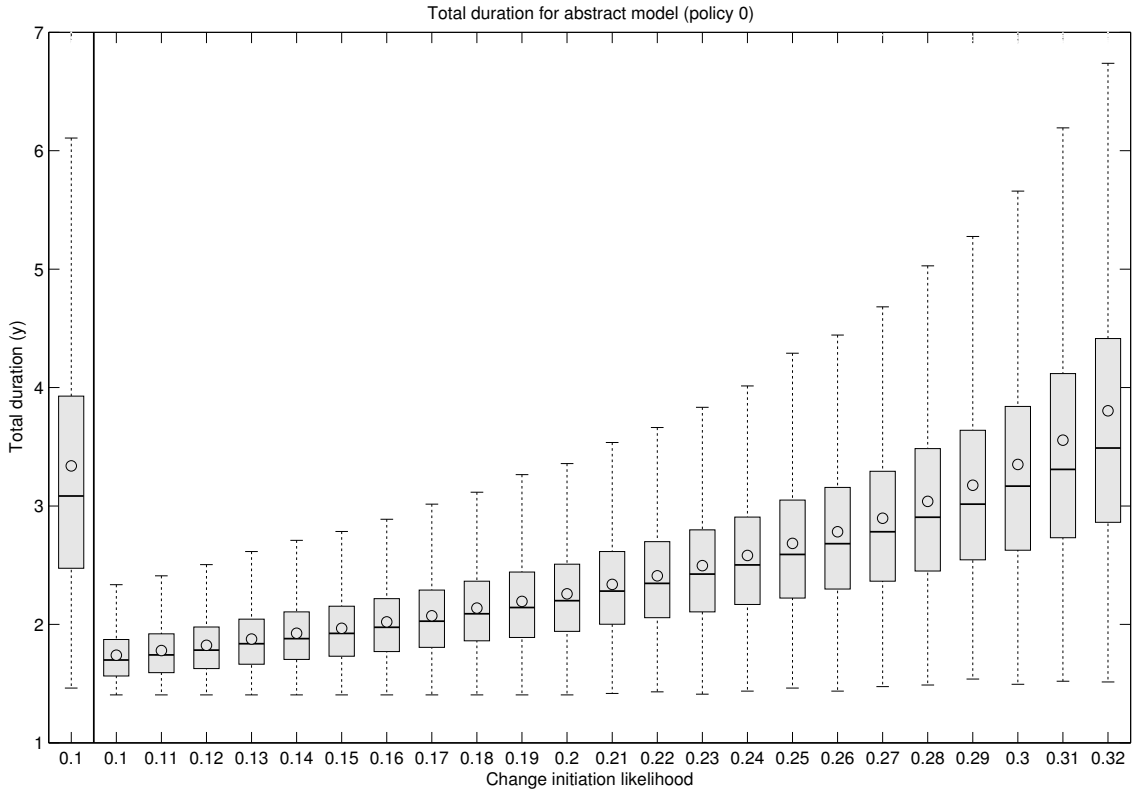

Figure 9: Sensitivity analysis for varying change initiation likelihood
(original model displayed on the left side)

Figure 10 displays simulation results of the aggregated model with adjusted p_c in more detail. It shows the relative policy performance and a histogram for policy 0, illustrating the shape of the distribution. A comparison to the results of the detailed model (Figure 8) reveals the decreased variance and impact of policies. It is worth noting that, while policy performance remains qualitatively similar, the effects of disadvantageous policies are underestimated.

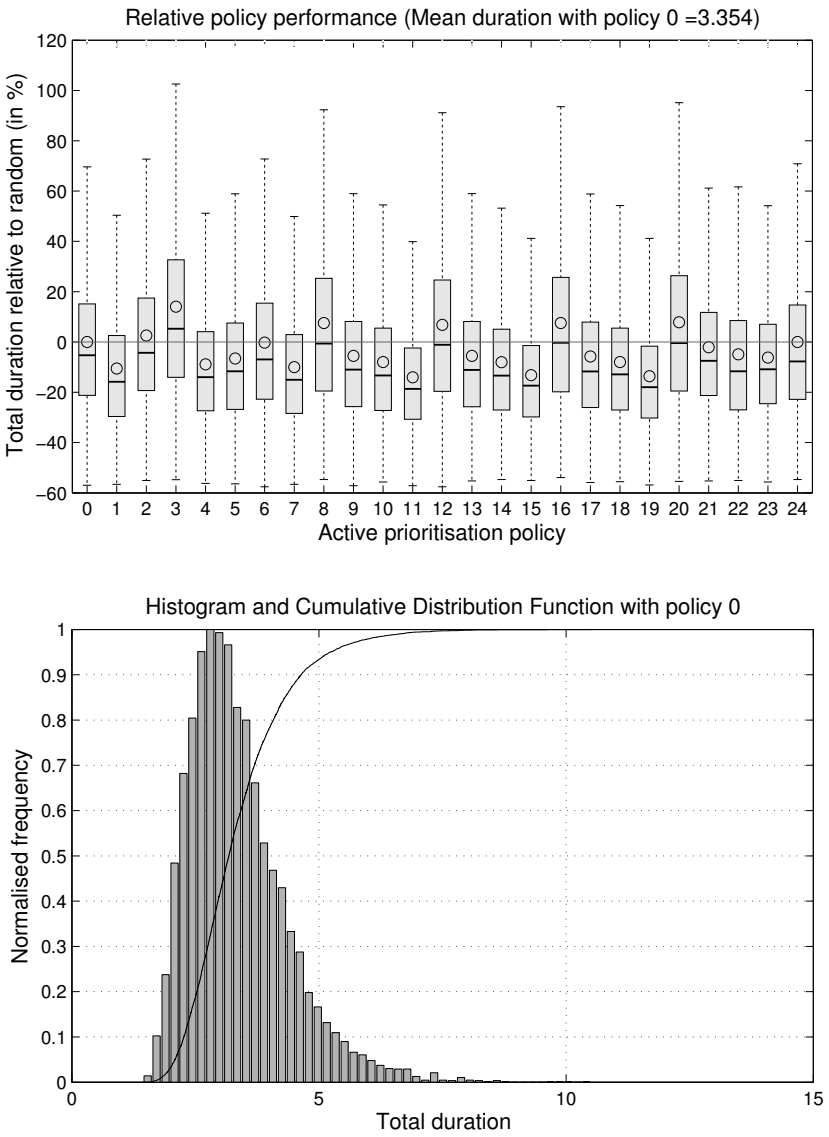

Figure 10: Simulation results for coarse-grained model with change initiation likelihood 0.3 (see also Figure 8 for a comparison with the original model)

b) Aggregation of task durations on system level

The durations for developing systems can be estimated by simulating the development of individual systems and assuming the average as the amount of time it takes to develop this system. This takes iteration and rework on an intra-system level into account, albeit in isolation for individual systems, and thus increases the information content of the assumed task durations. As Figure 11 shows, the difference to the original simulation with the coarse-grained model (Figure 8) - a 11% increase in mean task duration - is marginal relative to the results for the fine-grained model. While this includes the effects of intra-system change propagation, propagation paths crossing system boundaries are not accounted for on a component level.

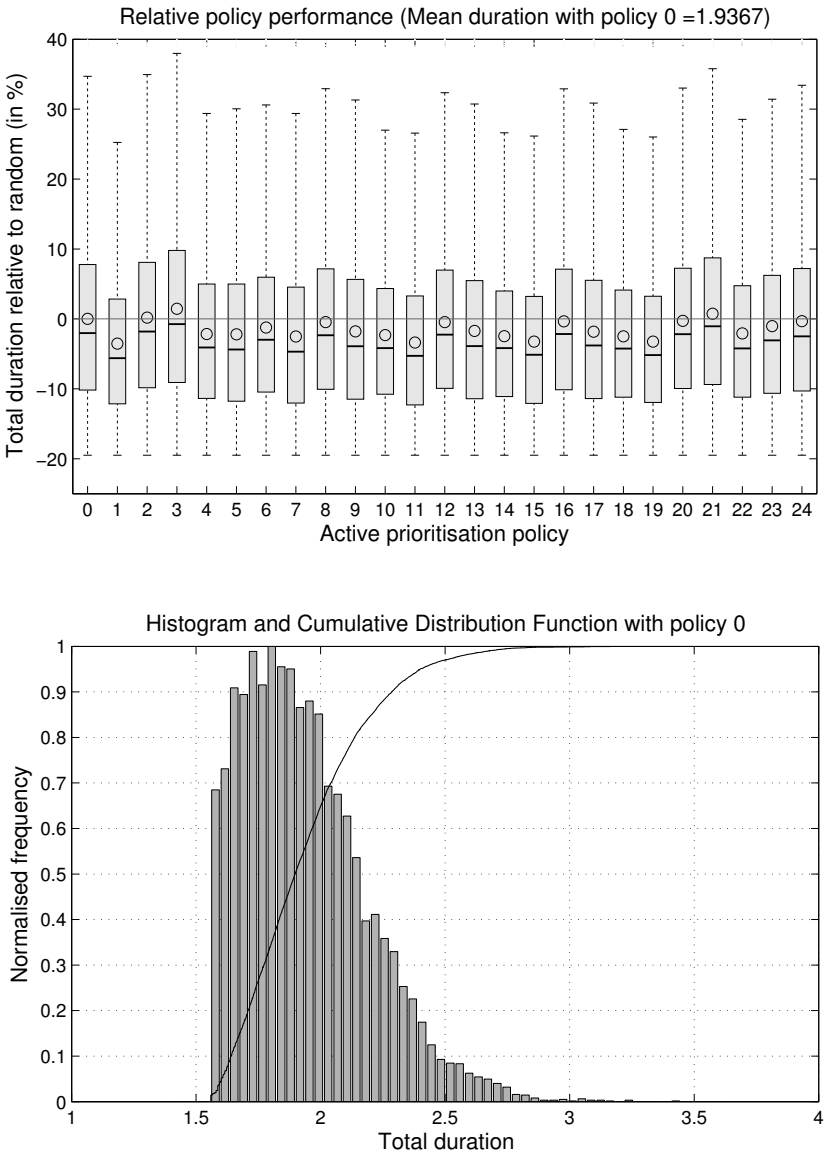

Figure 11: Simulation results for coarse-grained model with simulated system durations (see also Figure 8 for a comparison with the original model)

c) Number of maturity levels per system vs. component

Given that a system comprises multiple components, the number of maturity transitions to design a system is assumed to be higher than for individual components. This can be taken into account by increasing the respective model factor m. Since the fine model contains approximately four times as many components, the number of maturity transitions is increased fourfold, from 4 to 16, and the maximum allowed maturity difference (Δm_{max}) is adjusted accordingly, from 2 to 8. These adjustments lead to a finer simulation resolution and thus represent a trade-off between the structural granularity of the product model and the information granularity of the analysis. Figure 12 shows that this leads to a similar mean project duration as with the original fine-grained model, however with marginally less variance. Relative policy performance is less pronounced but remains qualitatively similar.

While a specific configuration was chosen, the results indicate that different types of granularity may be traded off to simulate design process performance. It has to be noted, however, that intra-system iterative behaviour is not directly captured, leading to fewer overall iterations with less variance. Also, due to the simulation mechanism, all systems have the same number of maturity progression steps, which may not appropriately represent their internal complexity.

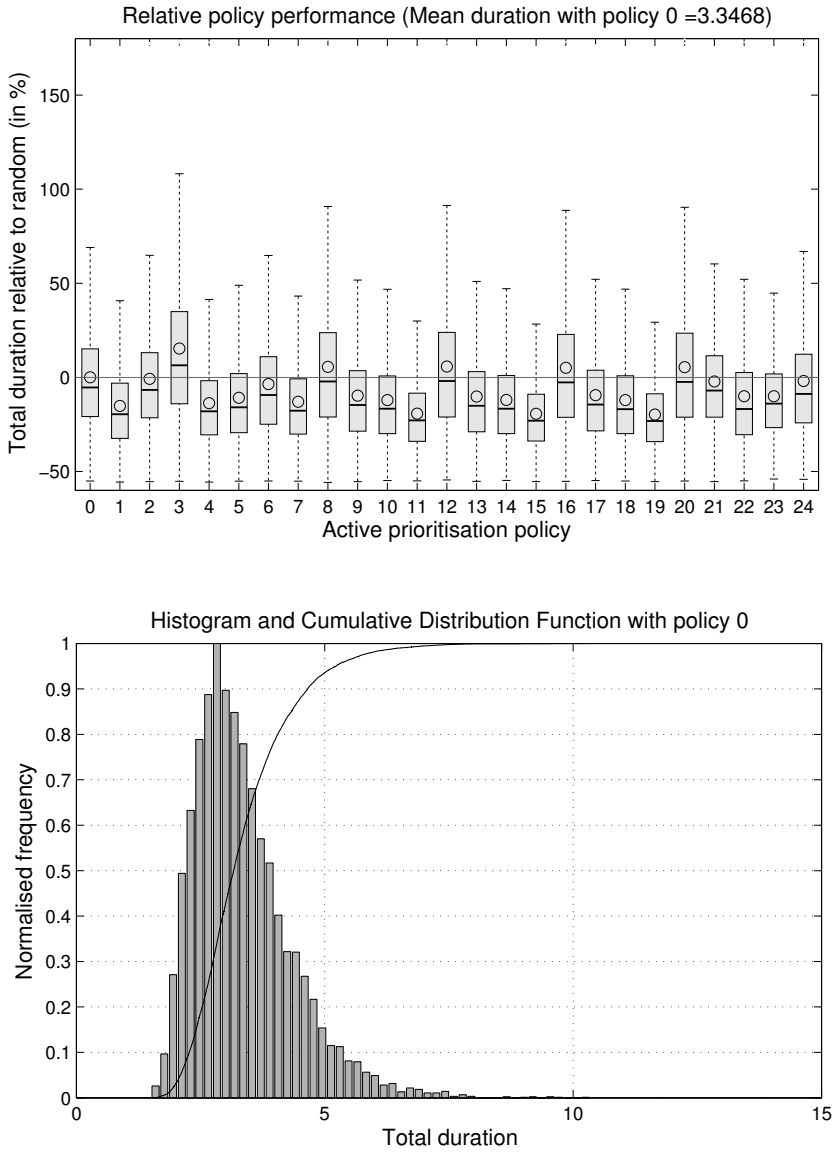

Figure 12: Simulation results for coarse-grained model with increased simulation resolution – 16 maturity level transformations (instead of 4, see Figure 8)

d) Impacts of change propagation on a system level

The impacts of change propagation on a system level may be higher than on a component level. While the model implicitly takes this into account by increased task durations on a system level, and therefore higher rework efforts, the impacts of change propagation in the underlying CPM model are assumed to be identical. While no validated approach exists,

Ariyo (2007) explores a way to aggregate impacts from a detailed model, noting the underlying conceptual difficulties. The impact only comes into effect in the last step of a change propagation chain, while the likelihoods are multiplied along it. The impact between systems thus also depends on the 'frequency' of changes propagating system-to-system. This can be accounted for by computing weighted averages, using likelihoods as weight factors (Ariyo 2007). Using this algorithm on the diesel engine model yields only two system-to-system dependencies with an increased impact (from a maturity reduction of one level to two). The differences in the simulation results are therefore marginal. Given that the aggregation of change propagation impacts, as opposed to likelihoods, is not validated and conceptually challenging, further work is necessary to obtain a valid aggregation algorithm. This would increase the information content of system-to-system relationships, thereby enabling to trade off coarser structural granularity with finer information granularity.

5.3 Implications of different granularities in the simulation model

The experiments conducted in this study illustrate the impact changes in (structural) granularity can have on the results of the investigated simulation model. Variations of model granularity have implications for a range of other model parameters and assumptions, some of which can also be used to mitigate the resulting effects. In particular, change initiation likelihood, assumed task durations, change propagation impacts, number of maturity levels, learning effects and task prioritisation rules can be adjusted to account for different structural granularities. These options, the underlying trade-offs and assumptions are discussed in this section. Figure 13 provides an overview of the simulation results of the conducted experiments to facilitate comparison.

The most straightforward way of adapting the simulation to different input model granularities is to adjust the overall change initiation likelihood. This follows the logic that changes are more likely to occur in a system, which comprises multiple components, than in a single component. While this allows to achieve a similar mean project duration and qualitative policy performance, it yields a more concentrated distribution and underestimates the effects of task prioritisation policies. The factor allows to scale the total simulated project duration but does not account for the concealed intra-system change propagation and the lower simulation resolution, leading to less variance in the results.

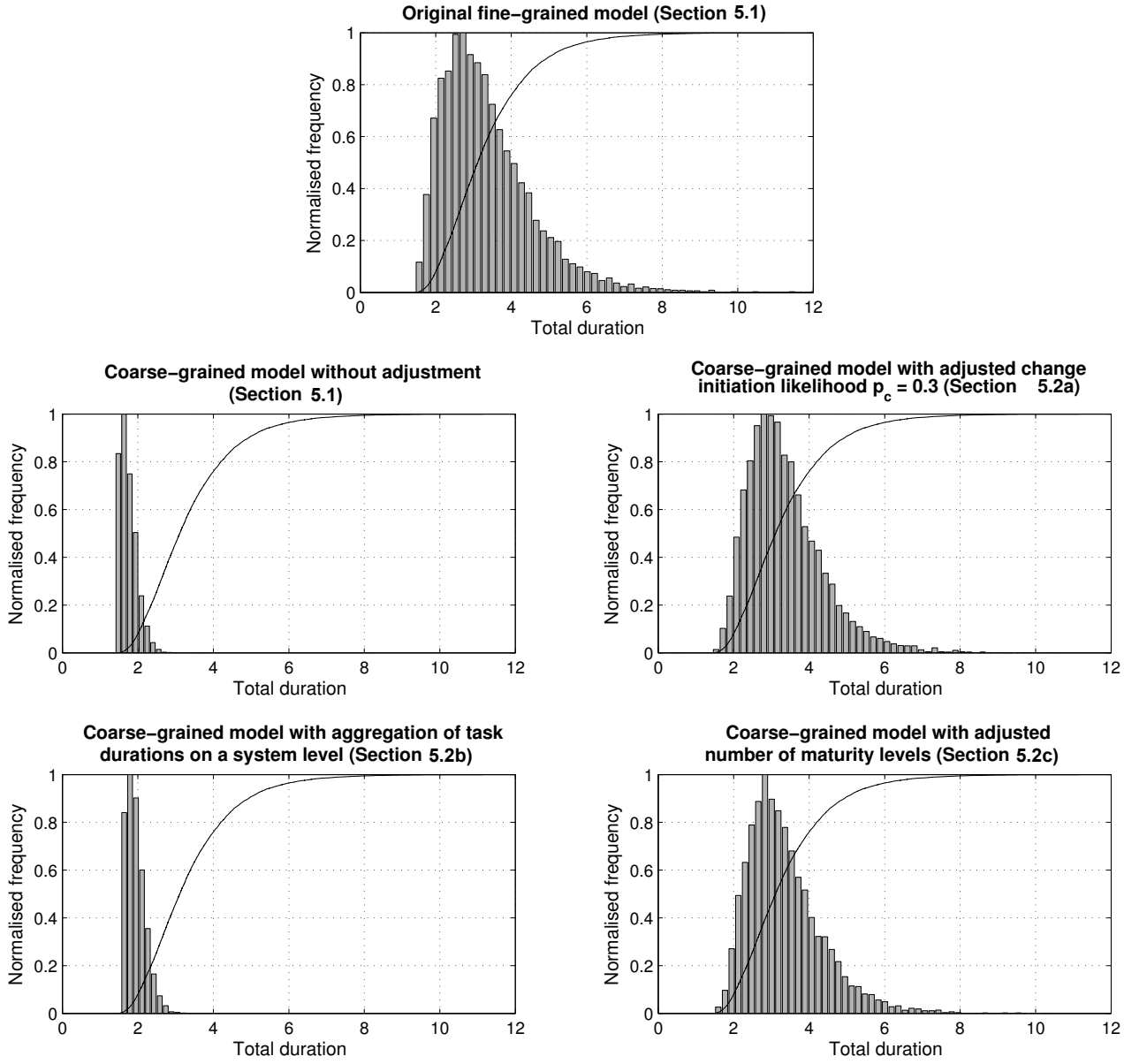

Figure 13: Overview of simulation results obtained in Section 5, displayed as histograms and cumulative distribution functions for 10,000 simulation runs with policy 0

Adding up the assumed durations to develop individual components does not account for intra-system iterations. This can be mitigated by simulating the development of individual systems independently and then using the respective mean durations as additional information to simulate the whole project. While this yields results that are closer to the original model, the project duration and policy performance are still underestimated. Another option would be to include a factor that accounts for intra-system iteration depending on system granularity.

In the original simulation model, components have a discrete maturity level between 0 and 4. Representing maturity progression more finely on a system level may be appropriate, given that systems comprise multiple components, each with specific design maturity levels.

Increasing the number of maturity levels according to the aggregation factor yields a total project duration similar to the fine-grained model. While variance and relative policy performance are still underestimated, this represents a way of trading off different dimensions of granularity by increasing the simulation resolution, thereby accounting for the coarser structural granularity of the model.

The algorithm used to aggregate the input model computes likelihoods but does not compute impacts of change propagation. Both values are used in the simulation model. This can be taken into account by calculating impact values in a similar fashion, which is conceptually more difficult – likelihoods are multiplied along a change propagation path while impacts only 'occur' in the last step. While the higher impacts on a system-level are partly accounted for by higher task- (and thus rework-) durations, predicting change impact on different levels remains challenging. Using an unvalidated algorithmic approach (Ariyo 2007) yields only marginally different impacts and therefore relatively similar simulation results. More research is necessary to develop mathematically and empirically viable multi-level impact estimation.

The learning curve in the original model, which is adapted from Cho and Eppinger (2005), may be more appropriate on a component level. While learning effects are also expected on a system level, their magnitude may be lower due to the more fragmented and distributed nature of developing bigger, more complex entities. Although priority rules are generic to development processes, they could be adjusted to account for granularity. Policies may be implemented, taking system size and intra-system connectivity into account. Moreover, the implications and purpose of such policies differ on less detailed levels, moving from task scheduling and operational planning to higher level strategic planning.

Simulation experiments with models on different levels of granularity reveal the impact on results as well as ways to account for this. While the conducted experiments only represent a fraction of conceivable configurations, focusing on one particular model, they do illustrate the implications of varying model granularity, provide an example how these can be studied and accounted for, and explore the potential trade-offs between different types of granularity. Even though some of the observed effects are specific to the used simulation model, they provide the basis for studying the implications of granularity more broadly. Based on the presented findings, Section 6 discusses how model granularity may affect analysis in more general terms and explores how such insights can provide guidance for modellers.

6 Limitations and further work

The experimental study presented in this article provides an indication of the implications of model granularity for design process simulation. It demonstrates that simulations are a promising way to analyse and demonstrate the importance of considering model granularity when building and using models.

Due to the scope and focus of this research, the analysis presented aims to provide an indication rather than a comprehensive account (insofar this is even possible) of the impact of granularity. Two models of a diesel engine on different levels of decomposition are considered, which allows for initial insights but not the deduction of a more general theory. Both the levels of granularity, number of architectures and different decompositions have to be increased to attempt comprehensive analysis. A bottom-up algorithm is adopted to aggregate based on a detailed model. Including top-down and combined perspectives could yield a better understanding of the role of granularity choices, but is conceptually more difficult to realise. For example, a model could be decomposed and aggregated again, analysing the effects and developing guidelines to ensure consistency. Because the chosen decomposition and change probabilities depend on the perspective of involved stakeholders, the influence of the original model on derivations with different granularities should also be investigated. The algorithm by Ariyo et al. (2007) is designed to conserve the structure of the models as much as possible in the aggregation processes, but it is possible that the dynamic behaviour of the models is different, as the connections between the aggregated will not cause iterations, which is likely to account for the denser distribution of the aggregated models.

The experiments explore four ways of adjusting the model to account for varying structural granularity of the input model (Section 5.2), modifying mainly information granularity. Extending this in terms of factors and granularity levels would help to quantify the relationships between model factors, input model properties and analysis results as well as to determine appropriate combinations. Another limitation concerns the two approaches used in this study (Maier et al. 2014; Ariyo et al. 2007). While they are thought to have face validity, given that they are synthesised from accepted and established concepts, they still only offer a particular perspective on the implications of model granularity. An extensive experimental study with a range of models on multiple levels of granularity is necessary to fully address these limitations.

This paper gives raise to many more detailed questions, that simulation of models at different levels of detail could answer such as

- How the strategies for priority rules are affected?
- Do maturity progression curves over time differ significantly between models with different levels of detail
- What general rules can be derived for modelling by looking at two different levels of detail
- Would the coarser model, be more argument if the probabilities are elicited from human experts?
- Would the results of the simulation be different, if a coarse model would have been refined rather than starting with a more detailed model?

Some of these questions would require artificially generated models on which statistical analysis can be carried out. However, this would go beyond the scope of this paper. In practise it is has also been proven very difficult to gain accurate change data (see Jarratt 2004) or to obtain models the process in detail in the case study company (Flanagan 2006), as the data is highly confidential and most likely capture in different formats based on different vocabulary. It doubtful that historic data could be captured over different product generations. Beside the time delay between generation, designers make a choice in each situation how to respond to a change which is highly context dependent (Eckert et al. 2005); and as designs have margins components and systems can absorb a certain amount of change in requirements, before turning from change absorbers into change multipliers (Eckert et al. 2004).

The research started with a fine grained model and generated a coarser model from it. This is a realistic scenario in industry in situation when an available product breakdown is too detailed for analysis. For example, the model used in Clarkson et al. (2004) started with a Bill of Materials breakdown and used two rounds of iteration to be make abstract enough that the model become usable. The practical problem in industry is however often that the granularity is uneven and that model builders need to aggregate components because they are very highly coupled and would generate spurious iterations in the simulation. Alternatively, company models might include whole systems as single entries, because they are bought in as a chunk, but the effect on subsystem is of interest.

Simulations models of processes would typically be built at the beginning of a process to gain insight into its behaviour. The designers using the models need to make a trade-off between the effort involved in building the models and the insights they can have through the model. Models are often reused between product generations or are used as a starting point in the modelling activity. The designers or analysts need to make a judgement whether the starting design is sufficiently similar to reuse a coarse or whether a greater level of detail is required to make this judgement about critical areas.

7 Conclusions

This article investigates the implications of varying granularity levels by reference to a DSM-based design process simulation model. An aggregation algorithm is used to create a change propagation model of a diesel engine on two different levels of granularity, which are used as inputs to conduct simulation experiments. This demonstrates the sensitivity of simulation results to shifts in model granularity, highlighting the importance of modelling choices regarding granularity. Based on the observations in this article, the ramifications of choosing and varying granularity for modelling more generally are discussed.

The main findings from the experimentation relate to the impact of granularity changes and potential ways to account for these. Changes in model granularity can have a significant effect

on analysis results. Abstract models may lead to results with less variance, underestimating delays due to iterations and impacts of policy interventions. In particular, more abstract models conceal lower-level effects, such as intra-system iterations, which may have significant influence on design process performance. Higher-level models may therefore misrepresent iteration behaviour across the model (Section 5.1). Depending on the modelling approach, model factors can be used to adjust model behaviour to granularity variations. Trading off different types of granularity (e.g. reducing structural granularity while increasing information granularity), may help to achieve an appropriate representation of a system or to retain model behaviour despite granularity variations (Section 5.2). By adjusting model factors, an aggregated model may yield average results similar to an underlying detailed model (e.g. project duration), which suggests that coarser models can also be used for initial planning purposes. However, the differences in the distribution of simulation results and relative policy performance indicate that the chosen granularity level has implications for analysis that cannot be predicted accurately (Section 5.2).

The simulation experiments presented in this article are a first step in determining the implications of model granularity for analysis and simulation. A comprehensive study is necessary to explore a wider range of modelling approaches and granularity variations as well as further representations of the simulation results (e.g. design progress over time). However, this article provides an indication of the impact of granularity as well as potential ways to study this and account for variations. In future research both the number of product architectures and levels of granularity should be increased and both bottom-up and top-down perspectives should be considered. Furthermore, the influence of the underlying original model on derivations with different levels of granularity could be explored. The effect of using models on different levels of granularity should also be studied for other modelling approaches. For instance, the presented experimentation could be extended to study other cross-domain cases with an underlying network structure, such as information flow processes on different hierarchical levels in an organisation.

References

Alexander C. (1964). *Notes on the Synthesis of Form*, Cambridge, MA: Harvard University Press.

AlGeddawy T. and ElMaraghy H. (2015). Determining Granularity of Changeable Manufacturing Systems Using Changeable Design Structure Matrix and Cladistics. *Journal of Mechanical Design*, 137(4), pp. 041702.

Ariyo O.O. (2007). *Change Propagation in Complex Design*. PhD Thesis, University of Cambridge.

Ariyo O.O., Keller R., Eckert C.M. and Clarkson P.J. (2007). Predicting change propagation on different levels of granularity: an algorithmic view. 16th International Conference on Engineering Design (ICED'07). Paris.

Braha D. and Bar-Yam Y. (2007). The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results. *Management Science*, 53(7), pp. 1127–1145.

Braha, D., & Maimon, O. (1998). A mathematical theory of design: foundations, algorithms and applications (Vol. 17). Springer Science & Business Media. Download at bit.ly/2MkU3uB

Braha D., & Maimon O. (1998). The Measurement of a Design Structural and Functional Complexity. IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 28(4), 527.

Browning T.R. (2001). Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management*, 48(3), pp. 292–306.

Browning T. R. (2002). Process integration using the design structure matrix. *Systems Engineering*, *5*(3), 180-193.

Browning T.R. (2016). Design structure matrix extensions and innovations: a survey and new opportunities. *IEEE Transactions on Engineering Management*, 63(1), pp. 27–52.

Browning T.R. and Eppinger S.D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Transactions on Engineering Management*, 49(4), pp. 428–442.

Browning T.R., Fricke E. and Negele H. (2006). Key concepts in modeling product development processes. *Systems Engineering*, 9(2), pp. 104–128.

Chiriac N., Hölttä-Otto K., Lysy D. and Suh E.S. (2011). Level of Modularity and Different Levels of System Granularity. *Journal of Mechanical Design*, 133(10), pp. 101007.

Cho S.H. and Eppinger S.D. (2005). A Simulation-Based Process Model for Managing Complex Design Projects. *IEEE Transactions on Engineering Management*, 52(3), pp. 316–328.

Clarkson P.J. and Hamilton J.R. (2000). "Signposting," a parameter-driven task-based model of the design process. *Research in Engineering Design*, 12(1), pp. 18–38.

Clarkson P.J., Simons C. and Eckert C. (2004). Predicting Change Propagation in Complex Design. *Journal of Mechanical Design*, 126(5), pp. 788–797.

Eckert C.M. and Clarkson P.J. (2010). Planning development processes for complex products. *Research in Engineering Design*, 21(3), pp. 153–171.

Eckert C.M., Albers A., Bursac N., Chen H.X., Clarkson P.J., Gericke K., Gladysz B., Maier J.F., Rachenkova G., Shapiro D. and Wynn D.C. (2015). Integrated product and process models: towards an integrated framework and review. Proceedings of the 20th International Conference on Engineering Design (ICED15). Milan.

Eckert, C., Clarkson, J., & Earl, C. (2005, January). Predictabilty of Change in Engineering: A Complexity View. In *ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (pp. 341-350). American Society of Mechanical Engineers.

Eckert, C., & Hillerbrand, R. (2018). Models in Engineering Design: Generative and epistemic function of product models. In *Advancements in the Philosophy of Design* (pp. 219-242). Springer, Cham.

Eckert C.M., Wynn D.C., Maier J.F., Albers A., Bursac N., Xin Chen H., Clarkson P.J., Gericke K., Gladysz B. and Shapiro D. (2017). On the integration of product and process models in engineering design. *Design Science*, 3.

Eckert, C. M., & Stacey, M. K. (2010). What is a process model? Reflections on the epistemology of design process models. In *Modelling and Management of Engineering Processes* (pp. 3-14). Springer, London.

Eppinger S.D., Joglekar N.R., Olechowski A. and Teo T. (2014). Improving the systems engineering process with multilevel analysis of interactions. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(04), pp. 323–337.

Eppinger S.D., Whitney D.E., Smith R.P. and Gebala D.A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1), pp. 1–13.

Flanagan T. (2006). Supporting design planning through process model simulation, PhD-thesis, Cambridge University Engineering Department.

Frigg R. (2009). Models and fiction. *Synthese*, 172(2), pp. 251–268.

Frigg R. (2003). *Re-presenting Scientific Representation*. London: PhD Thesis, London School of Economics.

Gericke, K., Eckert, C. M., & Wynn, D. (2016). Towards a framework of choices made during the lifecycles of process models. *Design*, *3*, 1275-1284.

Holschke O., Rake J. and Levina O. (2009). Granularity as a Cognitive Factor in the Effectiveness of Business Process Model Reuse. U. Dayal et al., eds. *Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 245–260.

Jarratt T.A.W. (2004). A model-based approach to support the management of engineering change, PhD-thesis, Cambridge University Engineering Department.

Jarratt T.A.W., Eckert C.M. and Clarkson P.J. (2004). Development of a Product Model to Support Engineering Change Management. Fifth International Symposium on Tools and

Methods of Competitive Engineering. Lausanne, pp. 331–342.

Jarratt T.A.W., Eckert C.M., Caldwell N.H.M. and Clarkson P.J. (2011). Engineering change: an overview and perspective on the literature. *Research in Engineering Design*, 22(2), pp. 103–124.

Karniel A. and Reich Y. (2009). From DSM-Based Planning to Design Process Simulation: A Review of Process Scheme Logic Verification Issues. *IEEE Transactions on Engineering Management*, 56(4), pp. 636–649.

Lévárdy V. and Browning T.R. (2009). An Adaptive Process Model to Support Product Development Project Management. *IEEE Transactions on Engineering Management*, 56(4), pp. 600–620.

Maier J.F. (2017). Granularity of Models in Engineering Design. PhD thesis. University of Cambridge.

Maier J.F., Eckert C.M. and Clarkson P.J. (2015). Different levels of product model granularity in design process simulation. Proceedings of the 20th International Conference on Engineering Design (ICED15). Milan, pp. 011–020.

Maier J.F., Eckert C.M. and Clarkson P.J. (2017). Granularity in modelling – concepts and framework. *Design Science*, 3.

Maier J.F., Eckert C.M. and Clarkson P.J. (2016). Model granularity and related concepts. International Design Conference - DESIGN 2016. Dubrovnik, Croatia, pp. 1327–1336.

Maier J.F., Wynn D.C., Biedermann W., Lindemann U. and Clarkson P.J. (2014). Simulating progressive iteration, rework and change propagation to prioritise design tasks. *Research in Engineering Design*, 25(4), pp. 283–307.

McMahon C.A. and Xianyi M. (1996). A network approach to parametric design integration. *Research in Engineering Design*, 8(1), pp. 14–32.

Pasqual M.C. and de Weck O.L. (2011). Multilayer network model for analysis and management of change propagation. *Research in Engineering Design*, 23(4), pp. 305–328.

Pidd M. (1999). Just modeling through: A rough guide to modeling. *Interfaces*, 29(2), pp. 118–132.

Robinson S., Nance R.E., Paul R.J., Pidd M. and Taylor S.J.E. (2004). Simulation model reuse: definitions, benefits and obstacles. *Simulation Modelling Practice and Theory*, 12(7-8), pp. 479–494.

Samy S.N., AlGeddawy T. and ElMaraghy H. (2015). A granularity model for balancing the structural complexity of manufacturing systems equipment and layout. *Journal of Manufacturing Systems*, 36, pp. 7–19.

Sarkar S., Dong A., Henderson J.A. and Robinson P.A. (2014). Spectral Characterization of

Hierarchical Modularity in Product Architectures. *Journal of Mechanical Design*, 136(1), pp. 011006–12.

Shannon C. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3), pp. 379–423.

Simon H.A. (1962). The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106(6), pp. 467–482.

Smith R.P. and Eppinger S.D. (1997a). A Predictive Model of Sequential Iteration in Engineering Design. *Management Science*, 43(8), pp. 1104–1120.

Smith R.P. and Eppinger S.D. (1997b). Identifying Controlling Features of Engineering Design Iteration. *Management Science*, 43(3), pp. 276–293.

Smith R.P. and Morrow J.A. (1999). Product development process modeling. *Design Studies*, 20(3), pp. 237–261.

Steward D.V. (1981). The Design Structure-System - a Method for Managing the Design of Complex-Systems. *IEEE Transactions on Engineering Management*, 28(3), pp. 71–74.

Suh E.S., Chiriac N. and Hölttä-Otto K. (2015). Seeing Complex System through Different Lenses: Impact of Decomposition Perspective on System Architecture Analysis. *Systems Engineering*, 18(3), pp. 229–240.

Wynn D.C. (2007). *Model-Based Approaches to Support Process Improvement in Complex Product Development*. PhD thesis, University of Cambridge.

Wynn D.C., Eckert C.M. and Clarkson P.J. (2006). Applied signposting: A modeling framework to support design process improvement. Proceedings of IDETC/CIE 2006. Philadelphia.

Wynn D.C., Eckert C.M. and Clarkson P.J. (2007). Modelling iteration in engineering design. 16th International Conference on Engineering Design (ICED'07). Paris.

Wynn D.C., Grebici K. and Clarkson P.J. (2011). Modelling the evolution of uncertainty levels during design. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 5(3), pp. 187–202.

Yao Y.Y. (2003). Probabilistic approaches to rough sets. *Expert Systems*, 20(5), pp. 287–297.

Yassine A.A. (2007). Investigating product development process reliability and robustness using simulation. *Journal of Engineering Design*, 18(6), pp. 545–561.

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

Yassine A.A., Whitney D.E. and Zambito T. (2001). Assessment of Rework Probabilities for

Simulating Product Development Processes Using the Design Structure Matrix (DSM). Proceedings of DECT. Pittsburgh, pp. 1–9.