Res Eng Design
DOI 10.1007/s00163-014-0174-8

ORIGINAL PAPER

Simulating progressive iteration, rework and change propagation to prioritise design tasks

Jakob F. Maier · David C. Wynn · Wieland Biedermann · Udo Lindemann · P. John Clarkson

Received: 22 June 2013 / Revised: 21 April 2014 / Accepted: 26 April 2014

Abstract Design tasks need to be rescheduled and re-prioritised frequently during product development. Inappropriate priority decisions generate rework; thus, the policy used to guide such decisions may have a significant effect on design cost and lead time. Generic priority rules provide easily implementable guidelines for task prioritisation and are theoretically effective for many planning problems. But can they be used in design processes, which include iteration, rework and changes? In this article, a discrete-event simulation model is developed to investigate priority policies in design. The model explores the combined effects of progressive iteration, rework and change propagation during design of interconnected parts in a product architecture. Design progression is modelled as an increase in the maturity of parts; rework and change propagation cause maturity levels in certain parts to reduce. Twelve product architecture models ranging in size from 7 to 32 elements are simulated to draw qualitative and general insights. Sensitivity of the findings to assumptions and model inputs is tested. Generally effective priority policies are identified, and their impact is shown to depend on the interconnectedness and organisation of product architecture, as well as the degree of concurrency in the design process.

Keywords Change propagation · Iteration · Process simulation · Design maturity · Design structure matrix

J. F. Maier (✉) · D. C. Wynn · P. J. Clarkson
Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK
e-mail: jfm45@cam.ac.uk

D. C. Wynn
e-mail: wynn@cantab.net

P. J. Clarkson
e-mail: pjc10@cam.ac.uk

W. Biedermann · U. Lindemann
Lehrstuhl für Produktentwicklung, Technische Universität München, Boltzmannstr. 15, 85748 Garching, Germany
e-mail: biedermann@mytum.de

U. Lindemann
e-mail: lindemann@pe.mw.tum.de

1 Introduction

The interconnected parts of a design emerge gradually together, through a process of incremental concretisation in which information is passed back and forth between teams working concurrently. There are usually many pending jobs but only limited resources to address them; thus, decisions must be made regarding which jobs have highest priority. In practice, these decisions cannot be fully planned out in advance, for instance, because unanticipated rework generated during the design process will disrupt the original plan. Priority decisions thus need to be adjusted in light of the changing situation as a project moves forward.

Considering these issues, this article sets out to:

1. Develop a model capable of exploring the combined effects of progressive iteration, rework and change propagation during design of intercoupled subsystems.
2. Use the model to determine whether task priority policies have a substantial effect on design process duration; and if so, whether generally applicable recommendations for task prioritisation can be identified.

Published online: 10 June 2014

The model rests on a design structure matrix (DSM) representing the structure of components and interfaces in the system being designed.[1] The state of design progress is represented as maturity levels that are associated with each component in the design. Undertaking work on a component ideally increases its maturity level, but sometimes reveals problems that reduce the maturity of the component, requiring rework to correct. The simulation accounts for knock-on effects of such maturity-reducing changes as they propagate because of the interfaces between components. The model combines the effects of three important design issues: (1) iteration that is undertaken to progress the design; (2) iteration needed to correct errors; and (3) change propagation due to interdependencies in the design.

The issues outlined above have been studied in existing research publications; however, a review reveals that they have not been synthesised and considered in combination (Sect. 2). The methodology for addressing Objective 1 above was therefore to construct a new model grounded in assumptions that are accepted in the literature, but to assemble them in a new way (Sect. 3).

To address Objective 2, simulation experiments were undertaken using the developed model (Sect. 4). A key consideration was to ensure that findings were generic, i.e. not unique to a particular design being studied. The analysis was therefore based on data collected from the literature. This was possible because the model is based on a product architecture DSM, which is widely used. Insights for management were drawn from the results (Sect. 5), and sensitivity analysis was conducted on the main assumptions to evaluate robustness of the findings. Contributions and implications are summarised in Sect. 6 prior to concluding in Sect. 7.

2 Background and related work

2.1 Progression in design

A design for a system or component is not completed in a single step, but requires a progressive process in which parameters are incrementally defined and frozen, and in which more accurate tools and analyses are gradually brought to bear as increased confidence in the design justifies the increased effort for their application. One way of perceiving design is thus as a gradual process of uncertainty reduction, in which the design description begins as a vague concept that allows for a wide range of final designs. This concept is concretised through a sequence of knowledge-generating activities and decision-making tasks, reducing the space of possibilities until a precise recipe for manufacture is reached (Antonsson and Otto 1995). According to this perspective, the aim of the design process is to generate knowledge that reduces uncertainty and increases design maturity.

There is a rich body of literature aiming to understand, represent and support the progressive nature of design processes. For instance, Antonsson and Otto (1995) distinguish between uncertainty, which represents uncontrolled stochastic variations, and imprecision, which is seen as uncertainty in choosing among alternatives. They propose the Method of Imprecision (MoI), based on the mathematics of fuzzy sets, for formal representation and manipulation of imprecision in engineering design. It is intended to support decision making and to facilitate coordination of concurrent engineering. Wynn et al. (2011) describe a process simulation model that captures iterative progression as uncertainty reduction in design, using the term 'uncertainty' to refer to everything that contributes to a lack of definition, lack of knowledge or lack of trust in knowledge. Their model considers the effects of design process tasks on up to five aspects of uncertainty associated with information that is iteratively developed during design: imprecision, inconsistency, inaccuracy, indecision and instability. The 'level' of each uncertainty aspect associated with a given piece of design information is modelled without reference to the information 'content' and is represented as a numeric value between predefined extremes. O'Brien and Smith (1995) consider progression in terms of design maturity, arguing that progress of immature designs should be prevented and mature designs should be progressed in concurrent design. They write that 'a design is mature when it is complete enough to allow the release of details to downstream activities, knowing that the release of further details from the current activity will not lead to redesign in any downstream activities. Knowing when a design reaches maturity would reduce the risk of releasing immature details to a following activity, and remove the delays caused by the need to check, through other methods, the validity of the design before its release' (O'Brien and Smith 1995). According to these authors, assessment of design maturity can prevent unnecessary delays and wasteful redesign. Grebici et al. (2007) also focus on maturity as a driver of design progress, presenting a framework to facilitate the exchange of knowledge about information maturity during design.

As hinted above, the fundamental process enabling progression in engineering design is iteration (Smith and Eppinger 1997b). Design is inherently iterative because of the cyclic interdependencies between parts of any design problem and because loops of synthesis, analysis and

[1] A DSM of a system comprising n parts is an nxn matrix in which each nonzero entry DSM_{ij} indicates that parts i and j are connected by an interface (Steward 1981; Eppinger et al. 1994). We use the term 'component' to represent either a subsystem or individual part in the design, according to the level of decomposition of the model.

evaluation are fundamental to the design process (Braha and Maimon 1997). Although iteration is therefore necessary to progress a design, it is also considered to be a major source of delays and budget overruns (Eppinger et al. 1994; Browning and Eppinger 2002). Smith and Eppinger (1997a) recommend two general strategies for accelerating an iterative design process: Faster iterations and fewer iterations. Both approaches require comprehension of the whole process, especially the coupling between its tasks. Considering the effects of hidden information in product development, Yassine et al. (2003) identify three main causes of churn leading to PD delays: interdependency, concurrency and feedback delays. They propose resource-based, rework-based and time-based strategies to mitigate these issues. In the context of complex product development networks, Braha and Bar-Yam (2004a, b, 2007) show that the propagation of defects or rework can be contained by prioritising resources at central components or tasks.

To summarise, modelling iteration is critical in any simulation of the design process (Wynn et al. 2007); ideally, the fundamental modes of progressive iteration and unnecessary rework should both be considered.

2.2 Change propagation

A high degree of interconnection between parts of a product leads to complex interactions during the design process (Eckert et al. 2004). In particular, whenever a component is changed during design (often due to iteration, although change can also arise from other sources), this can cause knock-on changes in other components so the design can continue to 'work together' as a whole (Lindemann and Reichwald 1998). This is termed *change propagation*. Change can propagate through different paths, depending on the connectivity between components of the product. It is thus important 'to be aware not only of individual change chains but of complex change networks' (Eckert et al. 2004). Almost all design activity includes engineering changes (Jarratt et al. 2011); Lindemann and Reichwald (1998) conclude from an extensive study that effective management of change propagation could provide a big advantage. This can be achieved through effective change prediction, which involves two activities: predicting the causes for change and predicting its knock-on effects (Eckert et al. 2004).

Numerous change prediction models have been proposed in the research literature, focusing mainly on understanding knock-on effects (Jarratt et al. 2011). One of the most established approaches is the Change Prediction Method (CPM) by Clarkson et al. (2004). The basic principle common to almost all the methods is that change can only propagate between two components if they are somehow connected. In CPM, for instance, the product architecture must be modelled as two DSMs, respectively, indicating the direct *impact* and *likelihood* of change propagation between each pair of components in the design. An algorithm accounts for the possibility that change may propagate between two components via several intermediate paths, thus increasing the possibility that the propagation may occur. Koh et al. (2012) build on CPM to develop a method which aims to support prediction and management of undesired engineering change propagation incorporating parameters and change options as well as component linkages. Another recent approach to change prediction is discussed by Yang and Duan (2012), who explore change propagation paths using a parameter linkage-based approach.

2.3 Design process simulation

Numerous articles have applied simulation to understand the impact of interdependencies and rework on the schedule and risk of design processes. Many are based on DSMs that represent the network of dependencies between tasks (Smith and Eppinger 1997b; Browning and Eppinger 2002; Yassine 2007). Approaches differ, among other things, in their simulation method, treatment of concurrency and treatment of task duration and rework (Karniel and Reich 2009). The main differences are discussed in the following subsections.

2.3.1 Simulation method

Karniel and Reich (2009) discern three main simulation methods: deterministic, Markov chain and Monte Carlo. Here, we distinguish between deterministic and stochastic models, where the latter category includes Markov chains, Monte Carlo methods, task- and agent-based simulations:

Deterministic models consider that the 'process progress is fully defined by its DSM structure' (Karniel and Reich 2009). For instance, Smith and Eppinger (1997a) consider the total work required to complete an intercoupled process using a deterministic model, in which the amount of rework created on each time step is related to the coupling strength between activities. Focusing on information transfer delays, Yassine and Braha (2003) use a similar logic and capture the fraction of rework created within a local group of tasks. Yassine et al. (2003) investigate information hiding in product development and extend the Work Transformation Matrix model of Smith and Eppinger (1997a) by accounting for different autonomous completion rates per component, reflecting developer's productivity levels. These are displayed along the diagonal of their improved Work Transformation Matrix. Abdelsalam and Bao (2006) discuss a deterministic model based on an 'iteration factor' representing coupling strength and the number of iterations required for convergence.

Stochastic models Huberman and Wilkinson (2005) base their work on the deterministic models by Smith and Eppinger (1997a) and Yassine et al. (2003) and include a stochastic component, which accounts for fluctuations in task performance and interactions between related tasks. Schlick et al. (2013) extend this further by formulating a vector autoregression model of cooperative work. Both recursive models incorporate inherent performance fluctuations.

Markov Chain models describe a process as a memoryless progression between states according to transition probabilities. For instance, Smith and Eppinger (1997b) develop a design process simulation using reward Markov chains generated from DSM sequencing. Sered and Reich (2006) also base a simulation on reward Markov chains of the sequential design process.

Monte Carlo models assume that a process behaves stochastically according to model-specific logic. In their task-based discrete-event simulation model, Browning and Eppinger (2002) apply Monte Carlo simulation to explore the impacts of activity sequence on cost and schedule risk in product development. Cho and Eppinger (2005) extend this approach by accounting for resource constraints and enhancing certain assumptions regarding task concurrency and rework.

Also falling within the category of stochastic models and in some instances using Monte Carlo techniques, recent engineering design literature has incorporated findings from research into complex engineered systems. For instance, Braha and Bar-Yam (2007) combine empirical and mathematical analysis of large-scale product development with stochastic simulation based on organisational networks. They model each node (task) in the PD network as having binary state, either resolved or unresolved. Stochastic rules are employed to evolve the state of each node based on its in-degree connectivity, the number of directly connected unresolved nodes and the internal completion rate of the node. Some findings from this class of model, which bear directly on the present article, are discussed in Sect. 2.4.

Agent-based models are another important class of stochastic design process model. They view agents as distinct information-processing entities whose actions are assumed to be rational responses to the limited information they receive, and the process emerges based on the resulting interactions between the agents. Garcia (2005) gives an overview of the use of agent-based modelling in product development. One established model in this category, the virtual design team, models the organisation, the work plan and their interactions to simulate micro-level information processing, communication and coordination behaviour of designers (Cohen 1992; Christiansen 1993; Levitt et al. 1999). Licht et al. (2007) propose a person-centred simulation model, incorporating the simulated product developer's bounded rational decision making. A recent contribution by Zhang et al. (2013) investigates local scheduling behaviour of designers and resource conflict resolution by managers in collaborative product development projects using agent-based simulation.

2.3.2 *Treatment of concurrency*

Models can be classified into four groups according to their treatment of concurrency (Karniel and Reich 2009). The first three groups comprise models that take a multipath approach; these execute activities fully in parallel (Smith and Eppinger 1997a; Yassine and Braha 2003), in parallel with possible overlap (Browning and Eppinger 2002; Yassine 2007; Krishnan et al. 1997; Loch and Terwiesch 1998) or serialised (Smith and Eppinger 1997b; Sered and Reich 2006). The fourth group is single-path approaches, which assume tasks are executed one-at-a-time (Lévárdy and Browning 2009).

2.3.3 *Task and rework durations*

The simplest approach here is to assume fixed, deterministic task durations that do not change on consecutive attempts (Smith and Eppinger 1997b). However, this may be oversimplistic (Smith and Morrow 1999). Many authors assume that rework of a task requires less time than its first execution. In the model reported by Browning and Eppinger (2002), for instance, task durations reduce on each consecutive iteration according to a predefined learning curve. Cho and Eppinger (2005) assume stochastic variation of task durations, which they write may also be combined with a learning curve effect.

2.4 Complex networks in product development

A range of techniques and models have been developed to increase understanding and enable the prediction of the behaviour of social, biological and technological networks (Newman 2003). Braha and Bar-Yam (2004a, b, 2007) show that large-scale product development networks have comparable structural properties and display similar statistical patterns to such systems. These structural properties can provide information on the dynamics of the product development process, for instance the characteristics of rework and its propagation. Braha and Bar-Yam (2004a, b) find that complex product development networks are dominated by some highly central tasks and characterised by an uneven distribution of nodal centrality measures and asymmetry between incoming and outgoing links. Braha and Bar-Yam (2007) consider priority rules based on the in-degree and out-degree of nodes and show that significant

performance improvements can be achieved by focusing efforts on central nodes in the design network. They see the 'close interplay between the design structure (product architecture) and the related organisation of tasks involved in the design process' (Braha and Bar-Yam 2007) as a potential explanation for the characteristic patterns of product development networks. Although their analysis focuses on organisational networks rather than product architecture, they argue that the statistical properties should be similar.

Braha and Bar-Yam (2007) develop a stochastic product development model and analysed its behaviour. They show that if you run the PD process on top of a random network, a threshold behaviour that depends on the average degree of the network determines whether the PD is stable or unstable, and how much time it takes.

In particular, they show that the dynamics of product development is determined and is controlled by the extent of (1) correlation among neighbouring nodes, and (2) correlation between the in-degree and out-degree of individual tasks. For weak correlations, a network with any topology was found to exhibit the same behaviour as a random network.

2.5 Resource-constrained project scheduling problem

Recalling that the objective of this article is to assess priority rules in design, another relevant area of research is the resource-constrained project scheduling problem (RCPSP). The objective is to minimise the lead time of a project by appropriately scheduling activities which are subject to precedence and resource constraints, but not iteration. Overviews of the many approaches to solving this problem are provided by Brucker et al. (1999) and Hartmann and Briskorn (2010). Because the problem itself is well defined, standard problem sets have been developed and widely used to compare the performance of different algorithms.

Commonly studied solution techniques include exact and heuristic algorithms as well as meta-heuristics. One of the most important solution techniques is priority rule-based scheduling (Kolisch 1996; Buddhakulsomsiri and Kim 2007; Chtourou and Haouari 2008). Browning and Yassine (2010) combine the resource-constrained multi-project scheduling problem (RCMPSP) with the use of task-based DSMs to analyse the performance of priority rule heuristics.

The RCPSP and RCMPSP literature demonstrates that priority rules provide a powerful approach to support managerial decision making. However, because these two problems do not include iteration or rework, it is not clear whether the insights are applicable to design processes.

2.6 Summary and critique

Progression in design cannot be achieved through a linear process, due to cyclic interdependencies between parts, parameters and interfaces in the system being designed. This necessitates an iterative process of progressively increasing maturity, in which work must begin based on incomplete or imprecise information, in the knowledge that this may later require corrective changes. Such changes propagate through the design via interfaces, creating rework. The rework must be scheduled and prioritised; inappropriate prioritisation may increase the total amount of work that needs to be done.

Models have been developed to simulate the design process considering different aspects of this behaviour and to generate insights for scheduling and prioritisation. Most task-based models capture the effect of rework, but do not directly account for change propagation due to structural connections within the developed product. Increasing progress of individual components' design maturity towards a final solution is usually not considered explicitly. Propagation effects have been extensively studied in change prediction models based on product architecture (component) DSMs; however, such models do not consider the time and cost of change completion as the process simulation models do. Finally, most scheduling algorithms consider the impact of priority rules but do not account for iteration and rework. No simulation model exists that explicitly combines all the following characteristics: rests on a component-based DSM; accounts for maturity progression in components; integrates the design process characteristics of iteration, rework and change propagation; and can evaluate the impact of task prioritisation in this context.

3 Model

A discrete-event process simulation model was developed to address the limitations summarised above. The model is based on a DSM of components in a design, in which numeric entries are used to describe the impact and likelihood of change propagation associated with each dependency. In common with other models in the literature [e.g. Clarkson et al. (2004)], it is assumed that the problem structure, as represented by this DSM, can be modelled in advance of the design process and does not change during it.

Maturity levels are assigned to each component to describe its state of progress, which changes during simulation. m discrete levels of maturity are used, allowing for $m - 1$ transformations between levels. Work on a component increases its maturity level. A decrease can be

triggered by initiated or propagated change—rework resulting from changes and their propagation is thus the main cause for delays in the simulated design process. The simulation starts with all components having the lowest possible maturity (0) and ends when all components reach their maximum maturity level ($m - 1$).

The algorithm is detailed in the following subsections, under headings that describe its three steps which repeat in a cycle until the simulated process is complete:

1. Identify task(s) to start.
2. Start task(s).
3. Complete task(s).

Model variables and the symbols used to refer to them are defined in Table 1.

3.1 Identify task(s) to start

The tasks that can be started at any time depend on the maturity levels of each component, availability of suitable resource and the relative priorities of any components that require the same resource. These aspects of the model are discussed in Sects. 3.1.1, 3.1.2 and 3.1.3 respectively.

3.1.1 *Accounting for maturity constraints*

The structure of the design, and the current maturity levels in each component, constrain the sequence of design progress. An example of constrained maturity level progression is a mechanical component moving from concept to 3D model to validated structural properties. In this case, it is not suitable to validate the part for stress distribution until the forces exerted by its opponents can be estimated. This, however, requires at least 3D models. So, the designs of all connected components have to be progressed to at least 3D models before stress analysis can be attempted.

Extending this argument, the model assumes that components which are interconnected in the DSM cannot be designed independently and must progress in lockstep. A component can thus only be selected for further work if the

Table 1 Model parameters

Parameter	Description	Definition
Model assumptions		
m	No. of maturity levels	$m \in \mathbb{N}$
Δm_{max}	Max. allowed maturity level difference	$\Delta m_{max} \in \mathbb{N} \mid \Delta m_{max} < m$
p_C	Probability of change initiation	$p_C \in \mathbb{R} \mid 0 \leq p_C < 1$
c_r	Change initiates in worked-on component (1) or any component (0)?	$c_r \in \{0, 1\}$
s_{max}	Max. no. of change propagation steps	$s_{max} \in \mathbb{Z}$
l_{min}	Minimum proportion of original task duration after learning effects	$l_{min} \in \mathbb{R} \mid 0 < l_{min} \leq 1$
l_s	Proportional reduction in task duration on each consecutive attempt	$l_s \in \mathbb{R} \mid 0 \leq l_s \leq l_{min}$
Design situation to be simulated		
n	No. of components in the design	$n \in \mathbb{N}$
q	No. of resources	$q \in \mathbb{N} \mid q \leq n$
$\boldsymbol{D}$	Duration to complete each component in absence of rework (days)	$\boldsymbol{D} \in \mathbb{R}^n \mid D_i > 0 \; \forall i$
$\boldsymbol{L}$	Likelihood DSM	$\boldsymbol{L} \in \mathbb{R}^{n \times n} \mid 0 \leq L_{ij} < 1 \; \forall i, j$
$\boldsymbol{I}$	Impact DSM	$\boldsymbol{I} \in \mathbb{Z}^{n \times n} \mid 0 \leq I_{ij} < m \; \forall i, j$
$\boldsymbol{Q}$	Component-resource mapping	$\boldsymbol{Q} \in \{0, 1\}^{n \times q}$
Priority rule to be evaluated		
f_i	Priority of task i (see Table 2 for priority rules studied)	$f_i : (\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{I}, \boldsymbol{M}, \boldsymbol{NR}, \boldsymbol{NT}) \rightarrow p, p \in \mathbb{R} \mid p \geq 0$
State variables which change during simulation		
t	Current simulation time	$t \in \mathbb{R} \mid t \geq 0$
$\mathbf{E}$	Queue of forthcoming task completion events (component and completion time)	$\mathbf{E} \in (C, t)^{n'} \mid C \in \mathbb{N} \mid C \leq n \mid n' \in \mathbb{N} \mid n' \leq n$
$\boldsymbol{W}$	Is each resource currently idle (1) or not (0)?	$\boldsymbol{W} \in \{0, 1\}^q$
$\boldsymbol{X}$	Is each component currently being worked on (0) or not (1)?	$\boldsymbol{X} \in \{0, 1\}^n$
$\boldsymbol{M}$	Current maturity level of each component	$\boldsymbol{M} \in \mathbb{Z}^n \mid 0 \leq M_i < m \; \forall i$
$\boldsymbol{NR}$	No. of times each component was reworked since last maturity increase	$\boldsymbol{NR} \in \mathbb{Z}^n$
$\boldsymbol{NT}$	No. of times each component has been attempted in total	$\boldsymbol{NT} \in \mathbb{Z}^n$

$\mathbb{Z}$ 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. In definitions of matrices and vectors, e.g. $\boldsymbol{Q} \in \{0, 1\}^{n \times q}$ indicates that $\boldsymbol{Q}$ is drawn from the set of all matrices having n rows and q columns, where Q_{ik} refers to the value in row i, column k, and each such value is drawn from the set $\{0, 1\}$

maturity levels of all other components it is dependent on are no more than Δm_{max} steps lower than its own. With X_i indicating whether component i is currently being worked on, M_i indicating the current maturity level of component i, and L_{ij} indicating the probability that a change to component j will propagate to cause change in component i, Θ_i determines whether component i is eligible for work (1) or not (0):

$$\Theta_i = \mathbb{1}_\alpha\left(\sum_j [1-\mathbb{1}_\alpha(L_{ij})]\cdot\mathbb{1}_\beta(M_i-M_j)\right)\cdot X_i\cdot[1-\mathbb{1}_\gamma(M_i)] \tag{1}$$

$$\alpha=\{0\};\quad \beta=\{x\in\mathbb{N}\mid \Delta m_{max}\le x<m\};\quad \gamma=\{m-1\}$$

In Eq. 1, the indicator function $\mathbb{1}_\alpha(x)$ returns the value 1 if x lies in subset α and the value 0 otherwise. The term $\left[1-\mathbb{1}_\alpha(L_{ij})\right]$ is a binary DSM whose entries are 0 if the corresponding entry in the likelihood DSM is 0 and 1 otherwise. $\mathbb{1}_\beta(M_i-M_j)$ indicates whether the maturity difference is less than Δm_{max} (returns 0), or not (returns 1). The product of these factors has to be 0 for all combinations of component i with components j it is dependent on. X_i indicates whether component i is currently worked on (0), or not (1). $[1-\mathbb{1}_\gamma(M_i)]$ indicates if component i has reached the maximum maturity level (0), or not (1).

Only if all these requirements are fulfilled, $\Theta_i = 1$ and component i is thus eligible for work. If no components are found eligible for work, the simulation is complete.

3.1.2 Accounting for resource constraints

Executing a design activity requires resource, which in this article is interpreted as the team or individual that can work on progressing the respective component. The model allows for two resource limitation schemes:

1. A resource is only able to work on certain specified components (thus simulating specialised skills of, e.g., control system engineers). This is denoted in the input matrix $\boldsymbol{Q}$. If resource k can work on component i, $Q_{ik} = 1$. Otherwise, $Q_{ik} = 0$.
2. Any resource can work on any component (thus simulating generic skills of, e.g. software developers on some kinds of project). In this case, $Q_{ik} = 1\forall i,k$.

An eligible component is only possible to progress at the current time if at least one resource is both idle and capable of working on it. With W_k defining whether a resource is idle, eligibility of component i accounting for both maturity level constraints and resource constraints may be written as Φ_i:

$$\Phi_i = \Theta_i\cdot min\left(\sum_k (Q_{ik}\cdot W_k),\ 1\right) \tag{2}$$

3.1.3 Making priority decisions

$\boldsymbol{\Phi}$ may identify more eligible components than there are resources available to work on them; a priority decision is required to account for such cases. The priority decision is simulated using a function that evaluates the priority for each eligible task; the task having highest priority is chosen to execute, and the others must wait. For each simulation reported in this article, one of 25 priority rules is used (Table 2). For binary DSMs, policies 9–20 are identical to policies 5–8, and for symmetric DSMs, active and passive sums are similar. The priority rules we test are adopted

Table 2 25 decision policies used in the simulation experiments

No.	Name	f_i	No.	Name	f_i
0	Random	1			
1	Lowest task duration	$1/(1+D_i)$	2	Highest task duration	D_i
3	Lowest maturity level	$1/(1+M_i)$	4	Highest maturity level	M_i
5	Lowest active sum (binary)	$1/(1+\sum_i(1-\mathbb{1}_\alpha(L_{ij})))$	6	Highest active sum (binary)	$\sum_i(1-\mathbb{1}_\alpha(L_{ij}))$
7	Lowest passive sum (binary)	$1/(1+\sum_j(1-\mathbb{1}_\alpha(L_{ij})))$	8	Highest passive sum (binary)	$\sum_j(1-\mathbb{1}_\alpha(L_{ij}))$
9	Lowest active sum (risk)	$1/(1+\sum_i L_{ij}\cdot I_{ij})$	10	Highest active sum (risk)	$\sum_i L_{ij}\cdot I_{ij}$
11	Lowest passive sum (risk)	$1/(1+\sum_j L_{ij}\cdot I_{ij})$	12	Highest passive sum (risk)	$\sum_j L_{ij}\cdot I_{ij}$
13	Lowest active sum (impact)	$1/(1+\sum_i I_{ij})$	14	Highest active sum (impact)	$\sum_i I_{ij}$
15	Lowest passive sum (impact)	$1/(1+\sum_j I_{ij})$	16	Highest passive sum (impact)	$\sum_j I_{ij}$
17	Lowest active sum (likelihood)	$1/(1+\sum_i L_{ij})$	18	Highest active sum (likelihood)	$\sum_i L_{ij}$
19	Lowest passive sum (likelihood)	$1/(1+\sum_j L_{ij})$	20	Highest passive sum (likelihood)	$\sum_j L_{ij}$
21	Fewest attempts of task	$1/(1+NT_i)$	22	Most attempts of task	NT_i
23	Lowest amount of rework	$1/(1+NR_i)$	24	Greatest amount of rework	NR_i

Variables and α defined previously

from Braha and Bar-Yam (2007) who derive them based on a strong empirical and theoretical basis. In the present article, these rules are tested in a more specific context, namely the progressive design of component structures through increasing maturity levels, occasionally set back by change initiation and propagation.

The selected decision policy f_i determines the priority of component i. The maximum priority among eligible tasks is thus:

$$\zeta = \max_i (\Phi_i \cdot f_i) \tag{3}$$

The set of highest-priority tasks competing for resource k may thus be written:

$$\eta_k = \arg\max_i (\Phi_i \cdot Q_{ik} \cdot f_i) \tag{4}$$

If no components are currently possible for resource k, η_k is an empty set. If multiple components have identical priority ζ, η_k contains multiple items. In this situation, a random tiebreaker is used to choose between the candidate tasks.

3.2 Start task(s)

After choosing the task to be started by each idle resource, as explained above, the durations are calculated and the model state updated accordingly. These steps are explained below.

3.2.1 Calculating task duration accounting for learning

The input $\boldsymbol{D}$ indicates the total time necessary to design each component. Task durations are deterministic (although stochastic variation could easily be added). Possible delays due to resource limitations, changes and rework are not included in this estimated total duration, because they are calculated by the model.

The time to complete each started task depends on its purpose:

1. Progressing between any two maturity levels for the first time is assumed to require a fixed and identical proportion of the total duration, ie. $\frac{D_i}{m-1}$ because there are $m-1$ possible transitions.
2. The duration of a task executed to account for change (thus, to return a component to a maturity level that had previously been reached) is subject to learning effects. This reflects the fact that it 'often [...] takes less effort to rework an activity than to do it the first time' (Browning and Eppinger 2002). The model assumes a linear relationship between the task duration and the number of rework iterations until a minimum proportion (l_{min}) is reached. This learning curve was selected to reflect the design process simulation model by Cho and Eppinger (2005), which is well established in the literature.

Taking both cases into account, when a task i is attempted during simulation, given that NR_i rework iterations have been completed to date, its duration Δt_i is:

$$\Delta t_i = \frac{D_i \cdot max(1 - l_s \cdot NR_i,\ l_{min})}{m-1} \tag{5}$$

3.2.2 Updating model state to start tasks

To recap, each entry in $\boldsymbol{\eta}$ contains a set of tasks (either one task or the empty set) to be started at the current time, by resource k.

For each resource k, the task i to be started (if any) is identified from η_k. The task is added to the event list $\mathbf{E}$ so that it will complete at time $t + \Delta t_i$. Resource k is flagged as working and component i is flagged as being worked on:

$$W_k = 0; \quad X_i = 0 \tag{6}$$

3.3 Complete task(s)

After starting the selected tasks, the simulation is advanced to the next event in $\mathbf{E}$ at which a task is due for completion. The model state is updated and the possibility of change initiation and propagation is considered. Once the task completion has been processed as explained below, the state has changed and new tasks may become possible to start. The model thus returns to the first step (Sect. 3.1).

3.3.1 Updating model state to complete tasks

The next event is removed from $\mathbf{E}$ and the simulation time t advanced accordingly. Resource k is flagged as idle, component i is flagged as not being worked on, the maturity level of component i and the total iteration counter for i are incremented:

$$W_k = 1; \quad X_i = 1; \quad M_i = M_i + 1; \quad NT_i = NT_i + 1 \tag{7}$$

If more than one completion event occurs at the same time, they are each processed in the same way.

3.3.2 Change initiation at task completion

Modelling occurrence of rework and changes as a random event occurring when an activity is completed is thought to be realistic (Browning and Eppinger 2002; Yassine et al. 2001). This model uses a similar logic to represent changes both internal and external to the product development process. In other words, when a change is initiated, it could be interpreted either as the task causing a change directly

or revealing a problem that may have originated elsewhere. If a change occurs in a component that is currently worked on, the activity is interrupted and the respective resource becomes available. The probability p_c of initiating change upon completion of a task is constant. The actions at task completion depend on whether a change is found to occur:

- If a change occurs, the component that is affected must first be determined. The model allows for two different assumptions here:
  1. Change occurs in the component just worked on (if $c_r = 0$);
  2. Change occurs in a component selected at random (if $c_r = 1$).

  After identifying the affected component, change is taken into account by reducing its maturity level:

$$M_i = \max(M_i - 1,\ 0) \tag{8}$$

  If a change occurs in a component that is currently worked on, this activity is interrupted.
- If a change does not occur and the component has just reached a new maturity high, the iteration counter is reset:

$$NR_i = 0 \tag{9}$$

3.3.3 Change propagation at task completion

When a change is initiated, it may propagate to other, directly connected, components. This is assumed to occur within the same time step as the initiated change. The rationale is that when dealing with design changes, companies typically follow a formal process in which the scope of the change is assessed prior to beginning work (Jarratt et al. 2012).[2]

When a change occurs, its effect in the simulation is determined using the change propagation model developed by Clarkson et al. (2004), which is well established in the research literature. The entries in the likelihood DSM $\boldsymbol{L}$ are consulted to determine the probability of change propagating from the initiating component to each component that is dependent on it. A Monte Carlo approach is used to determine whether each of these propagations occurs in the case at hand. For propagations that do occur, the process is repeated to identify second-order propagations, and so forth. Thus, the initiated change fans out through the product structure.

When a component i is subject to change propagated from component j, the maturity level is reduced according to the impact DSM entry:

$$M_i = \max(M_i - I_{ij},\ 0) \tag{10}$$

The more interconnected the product architecture and the higher the likelihoods of change propagation, the more extensive the resulting propagation tree is thus likely to be for any given change (see Clarkson et al. (2004) for further discussion of this property of the adopted propagation model; see Braha and Bar-Yam (2007) for application of similar logic to study organisational networks). Each branch is terminated when it exceeds s_{max} propagation steps, because in practice a change would not (be allowed to) propagate indefinitely (Pasqual and Weck 2012; Clarkson et al. 2004). It is further assumed that a change does not continue to propagate if the algorithm revisits a component that has already been selected for a change in the current timestep of the simulation. This reflects the logic of the propagation model reported by Clarkson et al. (2004).

3.4 Model summary

The model integrates existing concepts in the literature, namely iteration as a progression between maturity levels, rework initiation as an uncertain event that occurs on the completion of design tasks and propagation of initiated changes through the design according to likelihood and impact values. The model can be classified using the scheme referred to in Sect. 2.3 (Karniel and Reich 2009). It uses Monte Carlo simulation methods to account for occurrence and propagation of changes. Concurrency is treated using a multipath approach to execute activities in parallel with possible overlap. The model assumes rework to require less time than the first execution and thus accounts for learning effects to otherwise deterministic task durations.

Table 3 summarises the simulation algorithm. As well as the likelihood-impact DSMs $\boldsymbol{L}$ and $\boldsymbol{I}$ and the resource mapping matrix $\boldsymbol{Q}$, a number of parameters defining model assumptions are required. Suggested values are provided in Table 4. Some of these values may be justified by reference to prior studies as shown in Table 4. Even where such justification is not possible, making assumptions explicit in this way is useful because it allows their impact to be evaluated using sensitivity analysis. This is done in Sect. 5.

3.5 Example

This subsection illustrates the model logic by presenting an application to a product model of the Westland Helicopters EH101. The input for the simulation is a DSM indicating

[2] This distinguishes our approach from rework propagation models, in which propagation occurs between the input of a task and its output, and a propagation network is spread out over time according to the durations of the affected tasks. Further research would be needed to determine the effect of this focus on change propagation, rather than rework, on the model results.

Table 3 Overview of the simulation algorithm for a single run; this is repeated many times for Monte Carlo sampling

1. Load input data ($\boldsymbol{D}, \boldsymbol{L}, \boldsymbol{I}, \boldsymbol{Q}$) and initialise model variables ($t, \mathbf{E}, \boldsymbol{W}, \boldsymbol{X}, \boldsymbol{M}, \boldsymbol{NR}, \boldsymbol{NT}$; Table 1)
2. Determine eligible components accounting for maturity constraints (Eq. 1) and resource constraints (Eq. 2)
3. If components eligible and resources allocatable: next task to be inserted in the queue of forthcoming task completion events **E** is determined by selected decision policy (Eq. 4; Table 2) and its duration is calculated accounting for learning effects (Eq. 5)
Otherwise: Check for next task completion event and go to 4.
4. Update current status for occurring event (advance simulation time, increase respective maturity level and counter for attempted tasks; Eq. 7)
5. Monte Carlo methods determine if an initiating change occurs
If change occurs: reduce maturity level accordingly (Eq. 8)
Account for change propagation (Sect. 3.3.3) and reduce maturity level according to impact DSM $\boldsymbol{I}$ (Eq. 10)
Otherwise: Go to 6
6. Check if design is complete (all components have reached maximum maturity)
If no: Continue simulation run and go to 2
If yes: Terminate simulation run and go to 7
7. Calculate process performance metrics (Sect. 4.3) and save simulation trace

The bold letters refer to matrices and vectors

Table 4 Suggested values for the model assumption parameters

Variable	Value	Rationale
m	5	A larger number of maturity levels give a more interleaved process; this does not have significant effect on process duration, but takes longer to simulate because more discrete events must be processed
Δm_{max}	2	Design progress on a component is assumed to be strongly constrained by progress in connected components (see sensitivity analysis in Sect. 5.2)
p_c	0.1	Calibration shows this value to give an amount of rework in line with empirical studies in the literature (see Sect. 5.3.1)
c_r	0	Work on a component seems more likely to reveal problems in that component than in some other component (sensitivity analysis shows the assumption made here to have limited effect; see Sect. 5.3.3)
s_{max}	5	Justified by Clarkson et al. (2004) using a theoretical analysis and a study by Pasqual and Weck (2012). Increasing s_{max} has little effect (because the probability of longer propagation chains is low) but reducing it reduces process duration (because less propagated change is generated)
l_s, l_{min}	0.25	The improvement curve is based on that used by Cho and Eppinger (2005), assuming an improvement by a certain percentage (25 %) on each consecutive attempt until a minimum percentage (25 %) of the original duration is reached

impact and likelihood of change propagation between components in the helicopter. In this context, the term component refers to a helicopter subsystem, such as hydraulics or transmission. To generate the likelihood and impact matrices, a workshop and interviews were held with company personnel [see Clarkson et al. (2004) for details]. For the present article, this information was supplemented with an approximate duration for designing each component,[3] thus completing the input data required for the simulation model (Fig. 1).

In this example, policy 6 [highest active sum (binary)] and the assumption parameter values in Table 4 are used. For illustrative purposes, three identical resources are used and assumed to be able to work on all components. Simulating 10,000 runs results in a histogram of total process duration (Fig. 2). Each bin of this histogram represents a number of possible process outcomes. One of these outcomes is plotted as a Gantt chart in Fig. 3. Figure 4 shows the performance of all policies relative to the mean of random task selection.

The Gantt chart reveals the mechanics of the simulation. Up to three subsystems can be worked on at the same time, because this is the number of available resources. The four green bars in every row reflect that maturity has to be increased to a new high four times to complete a component's design. If changes cause rework in a component, a previously accomplished maturity level has to be reattained. As explained in sect. 3.2.1, such tasks are subject to learning effects. This is visible in Fig. 3 as a gradual decline of task duration between two green bars. The chart also shows that rework due to propagated changes occurs more frequently than rework caused by initiated changes, because of the fan-out of change through the product architecture.

The order of task execution is determined by the priority policy and is independent of the sequence in the DSM. In

[3] The approximate durations were not collected during the original workshops. They were estimated by the authors, one of whom participated in the Westlands EH101 project, for the purposes of demonstrating the algorithm. It is important to note that the material in this section is intended only to clarify the simulation model's operation prior to the experiments in sect. 5. Numerical results in this section should thus not be taken to provide definitive insights into the EH101 case.

	Air conditioning	Auxiliary electrics	Avionics	Bare fuselage	Cabling and piping	Engine auxiliaries	Engines	Equipment and furnishings	Fire protection	Flight control system	Fuel	Fuselage additional items	Hydraulics	Ice and rain protection	Main rotor blades	Main rotor head	Tail rotor	Transmission	Weapons and defence
Air conditioning			0.2 1	0.2 1		0.2 1	0.2 1	0.2 1				0.2 1							
Auxiliary electrics	0.2 1		0.2 1	0.2 1								0.5 1		0.2 1		0.2 1		0.2 1	
Avionics	0.2 1					0.5 1	0.7 1		0.2 1	0.5 1	0.2 1	0.5 1	0.2 1	0.2 1	0.5 1			0.2 1	0.5 2
Bare fuselage	0.7 1	0.2 1	0.7 1		0.7 1		0.7 1	0.7 1	0.7 1	0.2 1	0.5 1	0.7 1	0.2 1	0.2 1	0.2 1		0.2 1	0.2 1	0.5 1
Cabling and piping	0.5 1	0.7 1	0.7 1	0.5 1		0.2 1	0.7 1		0.7 1	0.5 1	0.5 1	0.5 1	0.5 1	0.5 1				0.5 1	0.5 1
Engine auxiliaries							0.7 1			0.2 1									
Engines						0.7 1								0.2 2					
Equipment and furnishings	0.2 1			0.2 1							0.2 1	0.2 1	0.2 1						
Fire protection				0.2 1			0.2 2				0.2 1								
Flight control system			0.2 1			0.2 1	0.2 1						0.2 1		0.2 1	0.2 1	0.2 1	0.2 1	0.2 1
Fuel				0.2 1		0.2 1	0.7 1					0.2 1							
Fuselage additional items				0.2 1															
Hydraulics				0.2 1						0.2 1		0.2 1					0.2 1	0.2 1	0.2 1
Ice and rain protection							0.5 1								0.2 1		0.2 1	0.2 1	
Main rotor blades														0.2 3		0.2 3			
Main rotor head										0.2 1				0.2 1	0.2 1			0.2 1	0.2 3
Tail rotor										0.2 1				0.2 2	0.2 1			0.2 1	
Transmission							0.5 1			0.2 1			0.2 1			0.2 1	0.2 1		0.2 3
Weapons and defence							0.2 1	0.2 1											
Task durations (days)	135	135	180	180	60	90	180	90	60	180	60	135	135	60	180	135	135	180	135

Fig. 1 Impact and likelihood DSM of subsystems in the Westlands EH101 helicopter (Clarkson et al. 2004). The *upper* value in each matrix cell indicates the likelihood of change propagation and *lower* value its impact, i.e. by how much the maturity of an affected component would be reduced in case of change

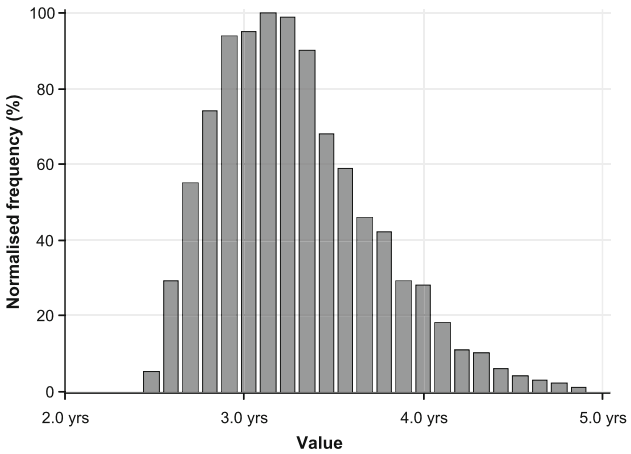

Fig. 2 Histogram of process duration resulting from 10,000 simulation runs of the EH101 model with policy 6

this case, under policy 6 the first component to be worked on has the most entries in its DSM column and the last component to be started has the fewest. For example, 'cabling and piping' and 'auxiliary electrics' both only have one outgoing relation to another component—'bare fuselage'. The design of both components is therefore started towards the end of the project. Because the design of 'bare fuselage' is dependent on the other components, it can only be advanced to a certain level once both predecessors have gained a certain maturity.

4 Analysis of priority rule performance

The simulation model introduced above addresses the first objective of this article, as stated in Sect. 1. To recap, the second objective was to use the new model to study the impact of priority rules on the performance of design processes incorporating iteration, rework and change propagation. To generate insights that are independent of a particular product and simulation configuration, a set of experiments was run using different configurations and input data.

4.1 Product models used in the experiments

The experiments used 12 product architecture DSMs drawn from the literature (Fig. 5). Each of these models was created through analysis of real products by the respective authors as listed in the figure caption. Three of the models include asymmetric input and likelihood data. The other nine models provide only symmetric binary DSMs. For simulation, these were transformed into impact and likelihood DSMs by assuming all likelihoods are 0.5 and all impacts are 1.

To study the influences of symmetry and sparsity as well as the sensitivity of the model to changes in the input data, additional input models were created based on the DSM of a chainsaw by adding entries. Ten or 20 entries were either randomly added to the upper and lower triangle (to create less sparse, less symmetric architectures) or symmetrically to both triangles (to create less sparse, more symmetric architectures).

4.2 Characterising the product models for comparison

To study the impact of a given priority rule across multiple product models, it is necessary to reduce the DSMs to a set of metrics that can be easily compared on one or more scales that ideally should be cardinal. Four metrics were used to characterise product models in the simulation experiments:

1. the number of items in the system decomposition (n), which is a direct input to the model.
2. the number of nonzero entries (NZ) in the likelihood DSM $\boldsymbol{L}$:

$$NZ(\boldsymbol{L}) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \left[1 - \mathbb{1}_{\alpha}(L_{ij})\right] \tag{11}$$

3. The nonzero fraction (NZF), which is the fraction of nonzero entries in $\boldsymbol{L}$, not counting the main diagonal.[4]

[4] This metric is referred to as *density* in graph theory and complex networks theory. It was earlier used and analysed in the context of large-scale product development by Braha and Bar-Yam (2004b), where it was shown that complex engineered networks are sparse, that is they have only a small fraction of the possible number of links.

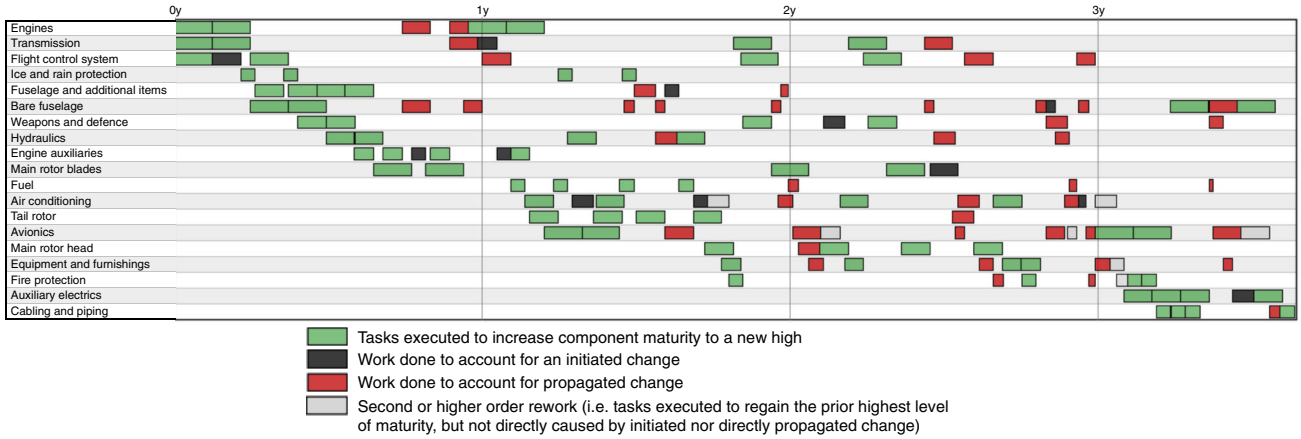

Fig. 3 Gantt chart showing the trace from an example simulation run on the EH101 model with policy 6

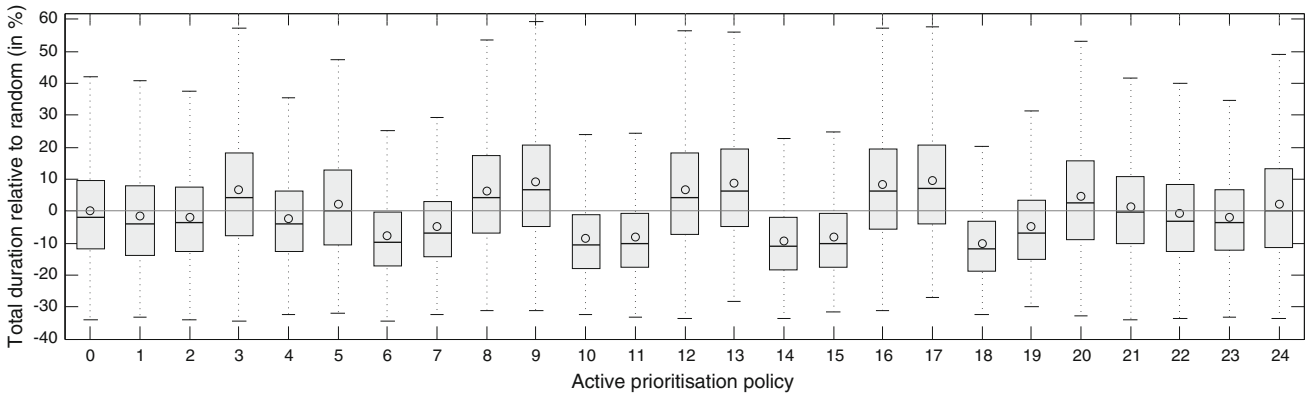

Fig. 4 Performance of all policies (relative to the mean of policy 0) for the EH101 model. The *bottom* and *top* of the *box* show the first and third quartile of each distribution, the band inside the box is the median and the circle is the mean. The ends of the whiskers represent the lowest (*highest*) datum still within 1.5 interquartile range of the lower (*upper*) quartile

It is thus a measure of the density of the DSM (Braha and Bar-Yam 2004b; Hölttä-Otto and de Weck 2007):

$$NZF(\boldsymbol{L}) = \frac{NZ(\boldsymbol{L})}{n(n-1)} \tag{12}$$

4. The symmetry factor (SF), which is the proportion of symmetric entries in all nonzero entries (NZ):

$$SF(\boldsymbol{L}) = \frac{\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |L_{ij} - L_{ji}|}{NZ(\boldsymbol{L})} \tag{13}$$

$NZF(\boldsymbol{L})$ and $SF(\boldsymbol{L})$ are normalised metrics with values in the range 0 to 1.

4.3 Characterising process performance

The sets of process traces resulting from simulation must also be comparable on ideally cardinal scales to evaluate policy performance under different conditions. The following metrics were used to compare performance of processes across simulation setups:

1. The total number of tasks attempted, considering every task that has been started, no matter if it was interrupted or requires less effort due to learning effects.
2. The total process duration, considering learning effects, idle times and interruptions.

Because the simulation results in a profile of possible processes, for each configuration 10,000 runs were used and the mean values of each metric calculated. The lower these two values are for a policy, the better its performance is assumed to be. Because of the simplifications made by the simulation model, relative performance is assumed to be more significant than the absolute results for total duration and tasks attempted. For each configuration studied, the performance of each policy is therefore presented as a percentage improvement or reduction in the metric values obtained from the random task selection policy for that configuration.

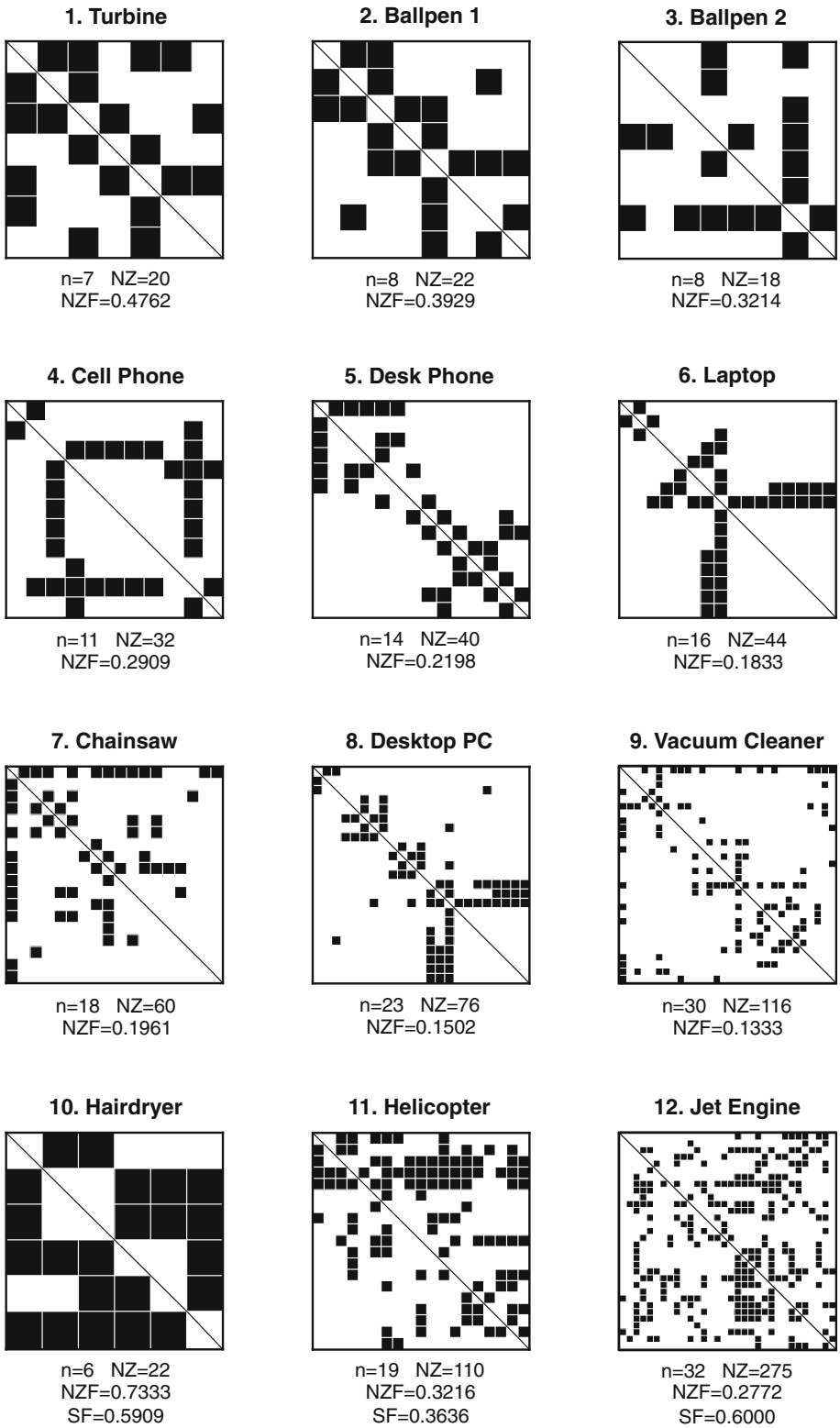

Fig. 5 Sparsity patterns of input DSMs. The models are based on a turbine (Maurer 2011), ballpen 1 (Maurer 2011), ballpen 2 (Lindemann et al. 2009), a cell phone, a desk phone, a laptop, a desktop PC (Hölttä-Otto and de Weck 2007), a chainsaw (Einögg 2009), a vacuum cleaner (Schmitz et al. 2011), a hairdryer, a helicopter (Clarkson et al. 2004) and a jet engine (Jarratt 2004)

4.4 Simulation experiments undertaken

The objective of the simulation experiments was to reveal general insights regarding whether the choice between policies in Table 2 has a significant impact on process performance, and what contextual factors influence which policy performs best. The factors that can be studied are determined by the parameters of the model, which were in

Table 5 Overview of the simulation experiments undertaken

	Product model (Figure 5)	Priority policy (f_i; see Table 2)	Number of resources (q)	Flexibility of resources (Q)	Probability of initiating change (p_c)	Max. change propagation steps (s_{max})	Max. allowed ML difference (Δm_{max})	Changes initiated in random location? (c_r)
§5.1.1	1-7 (plus DSMs with randomly added entries)	all	1	flexible	0.1	5	2	1
§5.1.2	6, 7, 11 (plus 7 with added entries and specifically created asymmetrical DSMs)	1-8, 23, 24	3	flexible	0.1	5	2	1
§5.1.3	2-6, 8	0, 5	1-5	flexible	0.1	5	2	1
§5.1.4	7	1-8, 23, 24	3	flexible or inflexible	0.1	5	2	1
§5.2	7	1-8, 23, 24	3	flexible	0.1	5	1-4	1
§5.3.1	11	0	3	flexible	0-0.2	0-5	2	1
§5.3.2	11 (original and with equalised entries)	1-8, 23, 24 (plus 9-20)	3	flexible	0.1	5	2	1
§5.3.3	7	1-8, 23, 24	3	flexible	0.1	5	2	0, 1

Rows indicate experiments and columns show the simulation parameters, with the shaded cells indicating the focus of each experiment

turn constructed from important issues identified during the literature review. Considering the available parameters, the following questions were identified for analysis:

1. Impact of design situation on policy performance
   (a) Does policy performance depend on size and interconnectedness of the design? (Sect. 5.1.1)
   (b) Given that some product models found in the literature are symmetric and some are asymmetric, does policy performance depend on the degree of symmetry of the product architecture? (Sect. 5.1.2)
   (c) Does policy performance depend on the number of available resources? (Sect. 5.1.3)
   (d) Does the flexibility of resources affect policy performance? (Sect. 5.1.4)
2. Impact of progressive iteration
   (a) How is policy performance affected by the maximum maturity level difference prior to starting a task? (Sect. 5.2)
3. Impact of rework and change propagation
   (a) How is policy performance affected by the probability of change initiation? (Sect. 5.3.1)
   (b) How is policy performance affected by the likelihood and impact of change propagation? (Sect. 5.3.2)
   (c) Does policy performance depend on whether change initiates in the completed task only, or in any task? (Sect. 5.3.3)
4. Overall policy performance
   (a) Can policies that perform well in most situations be identified? (Sect. 5.4)

Each of these eight questions was investigated through systematic variation of the simulation parameters followed by in-depth study of the outcomes. Table 5 summarises the experiments that were undertaken. In each case, 10,000 runs were performed using the indicated combinations of parameters and inputs. For most experiments, all input models and policies were initially included, although Table 5 only lists the configurations that are used to present the findings.

5 Results and implications

5.1 Impact of design situation on policy performance

5.1.1 Size and interconnectedness of the design

Question 1(a): Does policy performance depend on size and interconnectedness of the design?

The design of seven products modelled using DSMs of different size and interconnectedness was simulated to explore the impact of these factors. The effect of higher

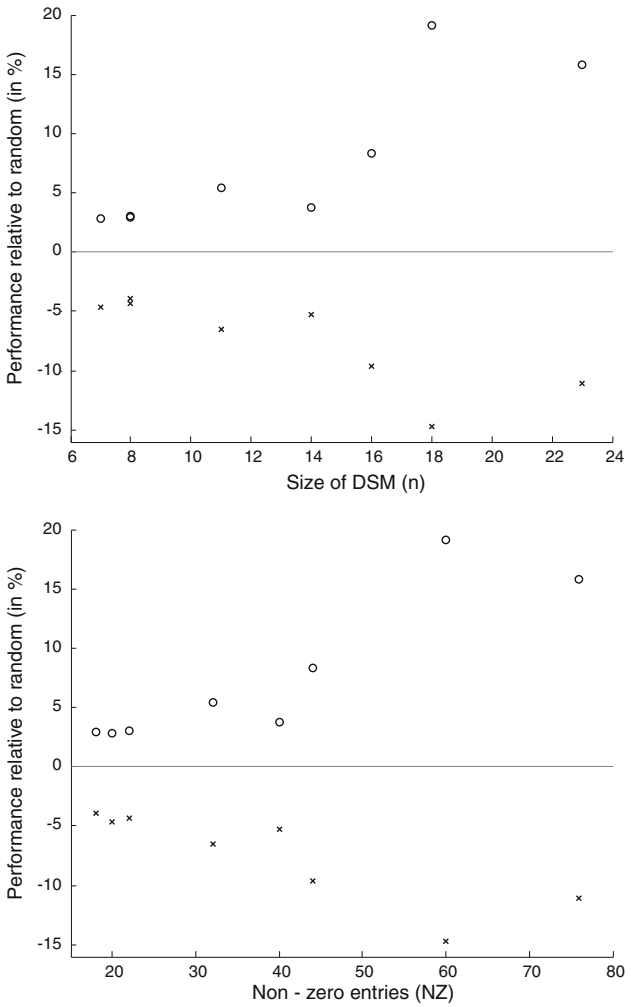

Fig. 6 Average simulated performance of selected policies relative to random (*O* lowest ML first, *X* highest ML first) according to *n* (*upper diagram*) and *NZ* (*lower diagram*)

nonzero fractions was further investigated by randomly adding entries to an existing DSM. All policies were tested and other parameter values for this experiment were as shown in Table 5.

Figure 6 shows the effect of *n* and *NZ* on the total duration for two policies relative to random task selection. Figure 7 shows the total simulated duration under the random policy.

These figures illustrate the general trend that higher values for *n* (system decomposition size) and *NZ* (connectivity) lead to greater process duration and variance and also amplify the relative differences in policy performance. In other words, the more decomposed and more interconnected a system design, the more important it is to take the right decisions regarding prioritisation of design tasks. This arises because products with many interrelated components are more susceptible to change propagation and thus create more priority decisions. The result confirms the more

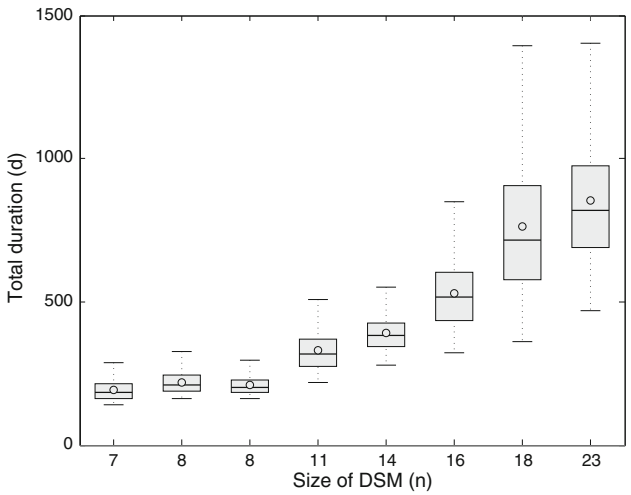

Fig. 7 Total simulated duration under random policy for DSMs with different *n*

general finding of Braha and Bar-Yam (2007) that the average degree of the network determines whether the PD is stable or unstable, and how fast it converges, although the mechanics of our model differ from their article. It also aligns with the findings of Yassine et al. (2003) who show, using different model mechanics again, that total PD duration decreases when the dependency strength between coupled components is reduced.

5.1.2 *Degree of symmetry of the product architecture*

Question 1(b): Does policy performance depend on the degree of symmetry of the product architecture?

To investigate the impact bi- and unilateral dependencies between components have on the total project duration, entries were either added to one triangle of a DSM or symmetrically to both triangles. To explore the effect symmetry has on the performance of various policies, two symmetric DSMs were compared with one asymmetric DSM and some specifically created, strongly asymmetric DSMs (for all other parameter values see Table 5).

For an identical number of added entries, the symmetric entries were found to result in a significantly higher total duration for most policies. This confirms the findings of Braha and Bar-Yam (2007), who use more general model mechanics to show that a high correlation between in-degree and out-degree of nodes in a task network can lead to instability and increased project lead time. A subset of results is plotted in Fig. 8. These show that the first two, symmetric DSMs give qualitatively similar performance whereas the third, non-symmetric shows differences for DSM-based policies. Results for DSMs with added asymmetrical entries confirm this trend and show more pronounced differences for lower symmetry factors.

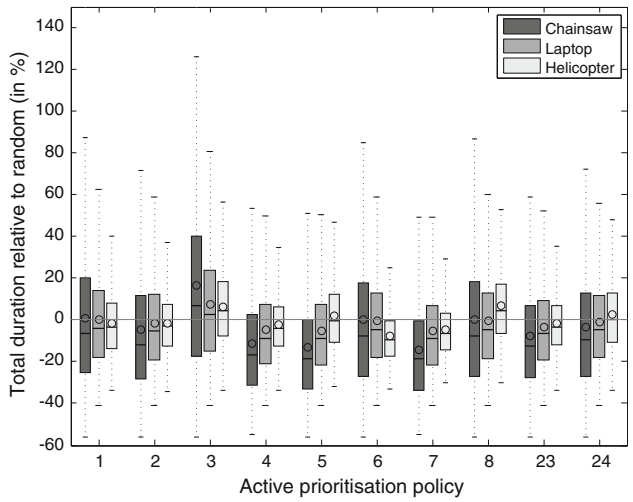

Fig. 8 Comparison of relative policy performance for three different product models and ten policies

In general, symmetric DSM entries have a higher impact on the results than the same number of asymmetric entries. It can be observed that symmetry and dependency structure affect performance of policies based on product DSMs, which is usually not the case for policies based on other factors.

Prioritising components with high active sums and low passive sums corresponds to prioritising the tasks in the order they would appear if the DSM was resequenced with the aim to reach a lower triangular form, such that components having high active sums and low passive sums would appear towards the top-left of the DSM. It can be concluded that, although the particular dependency structure being simulated has a strong influence on process performance, prioritising the component with the least passive dependencies is usually a good strategy.

5.1.3 Number of resources

Question 1(c): Does policy performance depend on the number of available resources?

To investigate the relation between the number of resources and the total duration to complete a design, four product models (3, 4, 6, 8) and two policies were compared for a range of one to five available resources. The influence of system size and degree of parallelisation was explored by comparing policy performance for six differently sized product models and a range of one to five available resources. All other parameters were as shown in Table 5.

Figure 9 depicts the relation of total working time and the degree of parallelisation. The total working time (total duration multiplied by the number of available resources) gives an indication of the effect of concurrency. It shows an increase in the total working time for higher numbers of available resources. This is more pronounced for the 'lowest active sum first' policy and for smaller systems. Figure 10 highlights this and shows a trade-off between the number of attempted tasks and the total duration for higher degrees of parallelisation and smaller systems.

It can thus be stated that the higher the number of resources, the less effect most policies have on the total duration. Specifically, for smaller systems and high degrees of parallelisation, trade-offs can occur between reducing the total duration and the number of attempted tasks.

While parallelisation generally leads to a faster completion of the design process, the total duration does not decrease by the same proportion the number of available resources increases. It is still possible to improve on random task selection, but the relative differences to random task selection are lower. The trade-offs that occur in smaller systems highlight the difficulty of scheduling design tasks for higher degrees of parallelisation and the importance of choosing the right degree of concurrency.

5.1.4 Flexibility of resources

Question 1(d): Does the flexibility of resources affect policy performance?

The experiments discussed in previous sub-sections assumed that any resource was capable of working on any task. This may be appropriate for some situations, such as software development, but does not reflect the organisation of most engineering companies into teams who have specific skills that are required to develop certain components. Sensitivity analysis was thus undertaken to compare the assumption of fully flexible resources to the situation in which specific teams are mapped to certain components they can work on. Simulations were run with the chainsaw product model for both assumptions. All other parameter values for this experiment are given in Table 5.

As shown in Figure 11, the total impact of this assumption on the policy performance is marginal, depending on the selected policy. The differences can affect magnitude while relative policy ranking remains very similar. However, it is notable that the mean total duration for random task selection is only about 4 % lower when resources are assumed to be identical. Similar behaviour was observed for other input models and variation of resource assignment.

The reason for this only marginal difference is that the impact of resource flexibility depends on the constraints on task sequencing, which in this case are governed by the maximum maturity level difference as well as the assignment of tasks across teams. If tasks can be attempted in many different sequences, which is the case when the maximum maturity level difference is significant, a

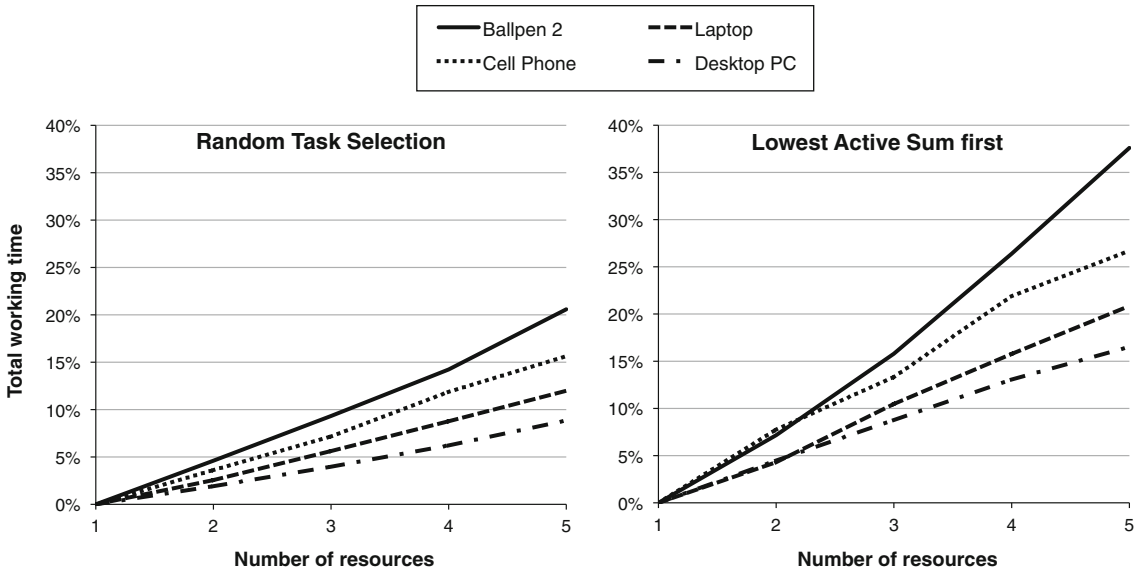

Fig. 9 Average simulated working time for an increasing number of available resources (1–5) and six input models. The percentages are based on sequential task execution

reduction in resource flexibility may not be overly constraining because a task sequence that avoids bottlenecks will often be possible. In the chainsaw simulation used to generate the figure, the 18 components are grouped into three groups of six components each. Each group of components can only be developed by one of the three teams. Thus, there remains a fair degree of flexibility in the simulated scenario.

5.2 Impact of progressive iteration

Question 2(a): How is policy performance affected by the maximum maturity level difference prior to starting a task?

The parameter Δm_{max} (see Table 4) constrains the activity sequence by specifying the maximum maturity level difference allowed between connected components for a task to be attempted. Using the chainsaw model, Δm_{max} was varied ($\Delta m_{max}: 1-4$) to investigate the effect maturity level constraints have on policy performance. All other parameter values for this experiment remained as shown in Table 5.

Figure 12 shows that in most cases, differences between mean policy performance increase while the variance of results decreases if Δm_{max} is assumed to be higher. In most cases, this has no effect on qualitative policy performance. It has to be noted that the absolute duration to finish the design increases with decreasing Δm_{max}. Based on $\Delta m_{max} = 4$ total duration for random task selection increases by 5, 13 and 21 %, respectively.

It can be concluded that the more strongly interfaces between parts constrain the activity sequence, the less the choice of policy affects design completion time. The choice between policies has the greatest impact if the activity sequence is not constrained at all. As the allowed maturity level difference between interconnected parts is reduced, policies become less important as they have fewer alternatives to choose among. In this case, the total duration to finish the design generally increases.

5.3 Impact of rework and change propagation

5.3.1 Probability of change initiation

Question 3(a): How is policy performance affected by the probability of change initiation?

The total duration to finish the design of the helicopter case was simulated using a range of values for the likelihood of change initiation ($p_c: 0-0.2$) and the number of change propagation steps ($s_{max}: 0-5$). All other parameters remained as shown in Table 5.

A subset of results is plotted in Fig. 13. This shows that if changes do not propagate, the total duration increases proportionally with the likelihood of change occurrence. The increase becomes incrementally more exponential when changes can propagate further. This effect weakens for four or five propagation steps. Depending on the simulation set-up, raising the likelihood for change occurrence eventually leads to a non-convergent behaviour in which the design cannot be finished. A similar increasingly exponential behaviour can be observed when the change propagation likelihoods in the DSM are all increased by the same proportion.

Figure 13 also shows that the total design time is about 80 % greater than the theoretical minimum completion

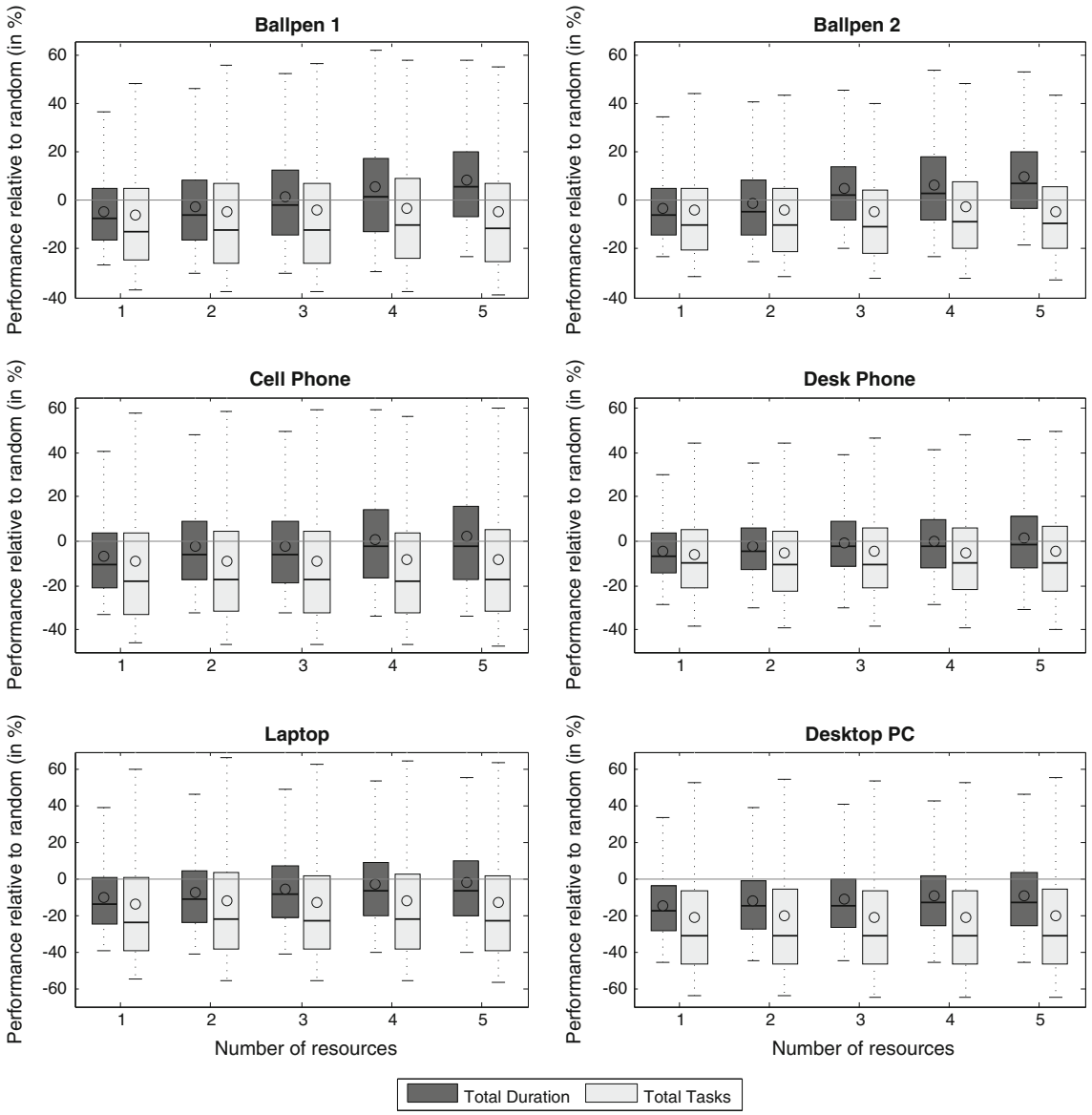

Fig. 10 Performance of 'lowest active sum first' (durations relative to 'random') for six input models depending on the number of available resources (1–5)

time for the helicopter model, given by the sum of all component durations. In other words, about 44 % of the total work is rework. Although the focus of the model reported in this article lies on generating qualitative insights about policy performance and not on predicting design process duration, comparison to prior studies suggests that these results lie in a sensible range. For instance, Chang (2002) found an average schedule increase of 69 % due to rework in four engineering projects. Ford and Sterman (2003a, b) report schedule changes of more twice the planned time, arising from rework. It can be concluded that assuming a 10 % likelihood of initial change occurrence gives sensible results and a model that is adequately realistic for the purpose of this article.

To summarise, the experiments in this subsection show that total duration and number of attempted tasks increase exponentially with higher likelihoods for occurrence of emergent changes and number of propagation steps. The former parameter was thus calibrated to give results in the region suggested by several prior studies, and the latter parameter uses the value from an earlier study reported by Clarkson et al. (2004).

5.3.2 Likelihood and impact of change propagation

Question 3(b): How is policy performance affected by the likelihood and impact of change propagation?

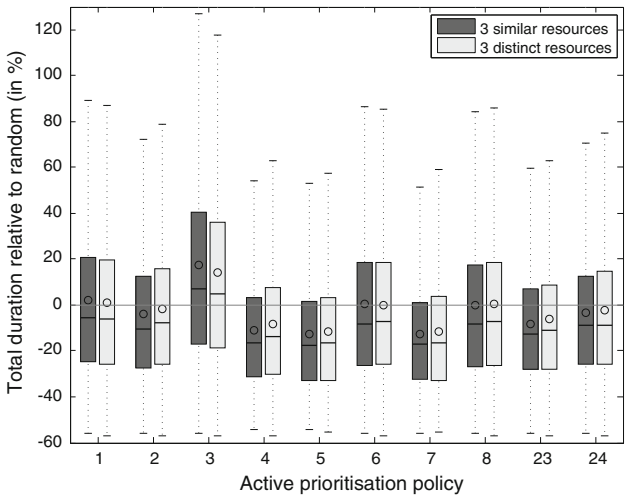

Fig. 11 Comparison of policy performance for the chainsaw product model. Simulations were run with (1) three identical resources that can work on all components and (2) three different resources that can only work on specific components

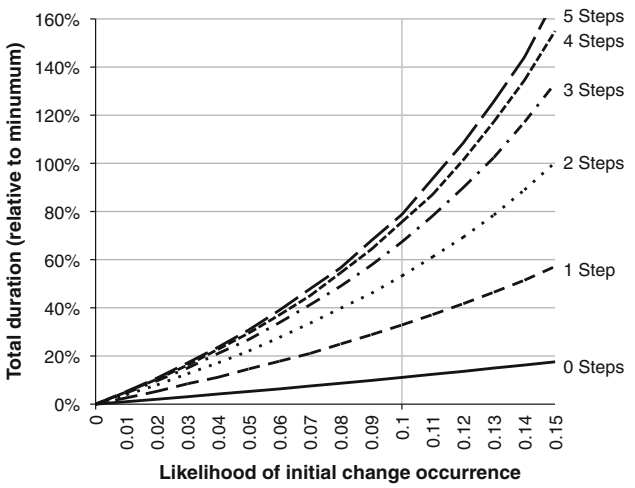

Fig. 13 Average simulated duration to finish design of the helicopter product model with random task selection for increasing likelihoods of initial change occurrence and numbers of change propagation steps

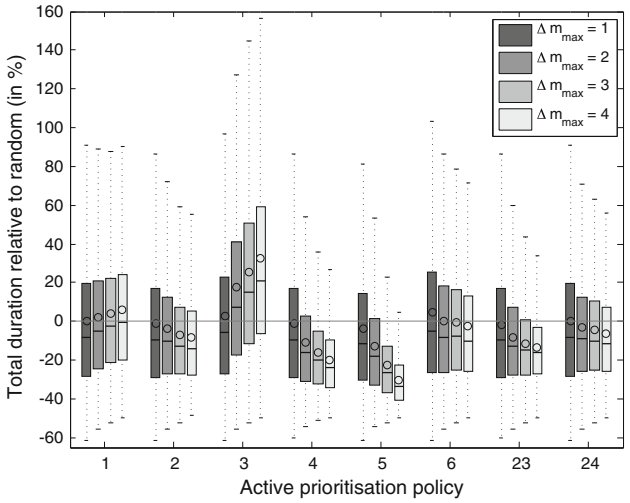

Fig. 12 Policy performance for different allowed maturity level differences in the chainsaw model

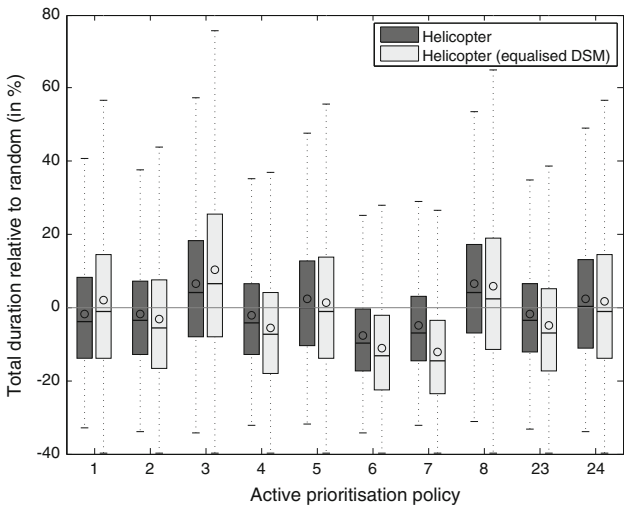

Fig. 14 Comparison of policy performance for the original product model with distinct impact and likelihood DSMs and an equalised version with identical entries across the DSM

The helicopter product model was simulated in its original state with distinct values for likelihood and impact of change propagation. This was compared to an equalised version that replaces all likelihoods with an average and all impacts with the value 1. See Table 5 for other parameter values.

Figure 14 shows little difference between the two set-ups. Policies that can generally be recommended perform well regardless of the type of input information. Even policies incorporating combined risk, impact or likelihood values show only marginal performance differences when applied to models where all impact and likelihood values are assumed to be equal. The variance of simulation results is slightly higher for the equalised input DSM.

Therefore, policy performance is qualitatively similar if the input model or the policy is based on binary or impact and likelihood DSMs. For the analysis reported in this article, it was not critical to elicit detailed dependency strength information.

5.3.3 Location of change initiation

Question 3(c): Does policy performance depend on whether change initiates in the completed task only, or in any task?

The results reported in previous subsections assumed that changes only initiate in the completed task (i.e.

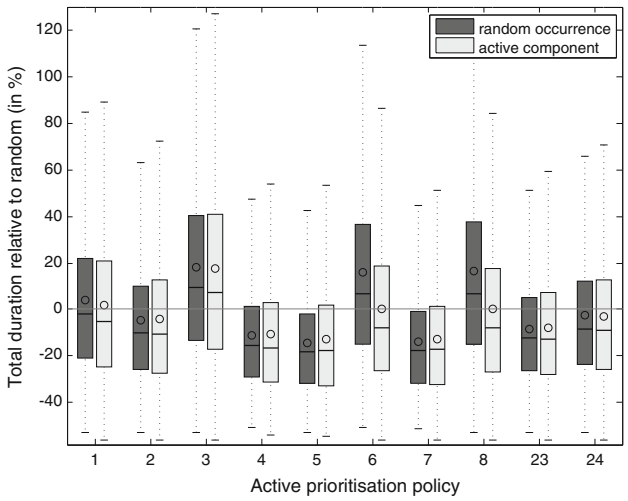

Fig. 15 Comparison of policy performance for the 'chainsaw' model. Simulations were run with changes either initiated randomly anywhere in the system or initiating in the task just completed

$c_r = 1$). To analyse the sensitivity of the model to this assumption, results were compared to the case where changes may initiate randomly anywhere in the system when a task is completed (i.e. $c_r = 0$). The chainsaw product model is used to illustrate this, while all other parameter values are as shown in Table 5.

Figure 15 shows that most policies are not strongly affected by the location in which changes initiate. Recommendable policies perform well either way. However, it is worth noting that for symmetric DSMs prioritising highest active/passive sums leads to similar results as random task selection if changes can only initiate in the component that has previously been worked on. If changes can occur randomly anywhere in the system, these policies lead to increased total durations.

5.4 Overall policy performance

Question 4(a): Can policies that perform well in most situations be identified?

Analysis of the results revealed the three most effective policies under a range of conditions to be:

1. Lowest passive sum first
2. Highest maturity level first
3. Least reworked first

These policies were found to be more effective than others tested across most of the scenarios studied. However, the specific situation does influence how well each policy performs. Therefore, this ranking is rather qualitative and should be understood as a suggestion of policies that perform well in most situations.

The performance of these policies can be explained in terms of the logic of the model. *Lowest passive sum first* can be interpreted as ensuring that tasks that could receive propagated change from many dependencies are not attempted until after those dependencies are finalised. *Highest maturity level first* and *least reworked first* can be interpreted as ensuring that the components which are thought to be least susceptible to change at any given point in the design process (because they have been reworked less than other tasks) are treated first. Work done to progress these components is less likely to be invalidated than work done to progress components that are historically more likely to receive change. These latter two policies may be valuable from a pragmatic point of view, because their use requires only knowledge of the history of the design process. Unlike *lowest passive sum first*, an explicit model of the product architecture is not required.

The least-recommended policy is *lowest maturity level first*. This policy always prioritises components that are lagging behind the rest. It may be interpreted as promoting firefighting, a phenomenon that often occurs in practice (Repenning 2001). The simulation results reported here confirm again that excessive firefighting may be detrimental to successful product development. The reason is that problems which keep occurring should usually be left until as late as possible, because repeated rework implies they are highly sensitive to changes elsewhere in the design.

6 Discussion

6.1 Summary and recap of contributions

This article makes two main contributions. Firstly, a new task-based model shows how design process simulation can account for the combined effects of progressive iteration, rework and change propagation in concurrent engineering. This is the first time these specific effects have been combined in a simulation study of the resource-limited design process. In future, widening the model's scope and combining its features with other approaches to simulating the design process could lead to a more realistic and thus more accurate representation of design practice.

Secondly, the model was used to study the impact of selected priority rules on design process performance, under different assumptions and for different products. The results, summarised in Table 6, show that certain priority rules are generally effective in reducing the impact of change propagation and unnecessary rework during design. Results also confirm that effective management of the jobs competing for attention becomes more important as the interconnectedness and concurrency of design projects increases.

Table 6 Summary of main observations and implications

	Main observations	Implications
Section 5.1.1	Absolute and relative differences in the results are more pronounced for bigger DSMs and higher nonzero fractions	Decisions become more critical in the development of strongly interconnected products. Decision policies can help to improve the outcome significantly
Section 5.1.2	Symmetric DSM entries have higher impact on the results than asymmetric entries. Symmetry and dependency structure affect the performance of policies that use active or passive sums as decision criteria	Mutual dependencies between components affect the design process more strongly than asymmetrical dependencies. It is important to keep dependency structure and symmetry-related issues in mind when choosing a DSM-based policy
Section 5.1.3	The higher the number of resources (i.e. the more parallelised the design), the less advantageous most policies are in terms of reducing the total duration. Specifically, for smaller systems and high degrees of parallelisation trade-offs can occur between reducing the total duration and the number of attempted tasks	A high degree of parallelisation can lead to performance losses of the policies because they do not regard the occupancy of resources or possible idle times. This affects especially simpler products, which makes it hard to choose the best policy in such scenarios
Section 5.1.4	Results vary only marginally according to whether resources are assumed to be able to work on any component or distinct and assigned to specific components	For the purpose of analysing policy performance in this simulation, the mapping of resources and tasks/components does not reveal major insights
Section 5.2	The stronger interfaces between parts constrain the activity sequence, the less effective policies are in reducing the design completion time	When prioritisation decisions have a broad scope of action, policies have a higher potential for improving the situation
Section 5.3.1	Total duration and number of attempted tasks increase exponentially with higher likelihoods for occurrence of emergent changes and number of propagation steps	The number of initiating changes has a big effect on the absolute results. The assumption made here leads to results in a reasonable range
Section 5.3.2	Policy performance is qualitatively similar regardless of whether the input model or the policy is based on binary or impact and likelihood DSMs	While the dependency structure remains crucial, for this simulation it does not seem worth the effort to elicit detailed dependency strength information
Section 5.3.3	In most cases, policy performance is qualitatively similar regardless of whether change is modelled as (1) initiating in the task just completed, or (2) initiated randomly anywhere in the system	The assumption of when and where emergent changes appear has an effect on the results albeit not a substantial one. This simplification seems to be acceptable for the purpose of this simulation
Section 5.4	The three most effective policies are: (1) *Lowest passive sum first* (2) *Highest maturity level first* (3) *Least reworked first*. The least-recommended policy is *lowest maturity level first*	These three policies perform well in most situations. Therefore, they can be recommended to serve as guidelines to help managers choosing and prioritising design tasks. Focusing attention on components that are lagging behind in terms of design maturity appears to be detrimental to performance

Several of the conclusions drawn from the simulation results are implied by prior theoretical and empirical results of Yassine et al. (2003) and Braha and Bar-Yam (2007), as noted in the text. The article thus confirms those prior results and their applicability to the specific context of engineering design as progression between maturity levels, set back by stochastic initiation and propagation of changes. The article also contributes new insights regarding progressive iteration through increasing maturity levels, a feature which is not explicitly incorporated in those prior models.

The box plots throughout this article reveal two insights about the variance in the results, which arises from the stochastic nature of the model. Firstly, the effect of the policies shifts the mean and also the whole distribution by an amount that is meaningful relative to the spread shown by the box. Secondly, in most cases, it is clear that the policies, which result in a reduction in time and effort, also result in a reduction in spread. Thus, the most effective policies can be expected to reduce variability (risk) as well as the expected (mean) performance. It may be noted that a differentiation of roughly 10 % in design process duration, as provided by comparing some of the policies studied in this article, can be significant in real terms. For instance it might imply avoiding a delay of 6 months on a 5-year aircraft program.

6.2 Comments on model validity

A key issue regarding any simulation model, and especially in cases where general insights are derived from simulation, is to understand the validity of the model and its input data.

In this case, the model is thought to have high face validity because it is synthesised from concepts that are well established and accepted in the research literature (Browning and Eppinger 2002; Clarkson et al. 2004; Cho and Eppinger 2005). The findings are developed directly from these assumptions, which might be refined in future studies but provide a reasonable basis for the current

article. Additionally, certain results of the simulation were found to reflect prior work in which more general models were used to analyse the impact of priority policies in PD (Braha and Bar-Yam 2007). This alignment of results with prior research further support the validity of the new model.

In terms of input data, the model was deliberately constructed in such a way that existing product DSM models can be used directly to provide most required data. Of the assumptions that are not captured in a product DSM, several of the required parameter values (maximum number of change propagation steps, average learning curve steepness, etc.) were sourced from previous widely cited research studies. Finally, the effects of most assumptions embedded in the model were evaluated through a series of sensitivity analyses. Most of these assumptions were found to have little or no impact on the qualitative results as presented in Table 6. A critical assumption was found to be the probability of change initiation. This value was calibrated by reference to existing reports on the amount of rework found in typical development projects.

The findings are therefore believed to be well justified and relatively robust, although there is still scope for improvement of the model and for further sensitivity studies. Some suggestions for enhancement are identified below.

6.3 Limitations and opportunities for further work regarding the model

The approach presented in this article, in common with all models, simplifies actual conditions in a number of ways. Some of these simplifications and opportunities for further work are discussed below.

6.3.1 Simplifications regarding maturity level progression

The model assumes discrete maturity levels, and each progression is assumed to require equal effort. In practice, it is likely that the effort required to reach a given maturity level will change as the process progresses and also that different components will have different numbers of natural maturity levels. The design of a component could be further divided into a network of tasks that involves engineers from several disciplines. Components may be designed concurrently, with designers transferring information in a less structured way than at maturity level transitions. The effects of these simplifications could be studied in future work.

6.3.2 Simplifications regarding resource constraints

In reality, the constraints regarding which personnel can work on which components are more complex than included in our model. For instance, highly specialised domain experts might constrain the task sequence and lead to bottlenecks. Also, personnel may have varying skill levels, which could affect the duration of tasks and the likelihood of creating rework. Further analysis, possibly including empirical study, could investigate extending the model to incorporate such issues.

6.3.3 Simplifications regarding the learning curve

The simulation uses a plateau learning curve model in which the task duration decreases linearly with the number of rework iterations until a minimum is reached. While this is an established approach in the PD simulation literature [see, e.g. Cho and Eppinger (2005)], the validity of this assumption can still be questioned. Further work could focus on adapting more sophisticated learning curve models, such as power functions or exponential functions (Anzanello and Fogliatto 2011).

6.3.4 Simplifications regarding the initiation of change

Following the logic of Browning and Eppinger (2002), changes are assumed to be initiated only on completion of a task. Further refinement of the model could incorporate the more realistic scenario in which changes initiate stochastically during task execution, rather than only at the end of the originally scheduled task. This could be expected to slightly reduce the negative impact of changes as calculated by the algorithm, because part of the initiating task would not need to be completed in case of change initiation. Consequently the effects of priority rules might be slightly less differentiated. Because most of the cost of change arises from dealing with propagation effects, not from completing the initiating tasks, this refinement would not be expected to significantly alter the results presented in this article.

6.3.5 Organisational and human behaviour issues

Inappropriate task sequencing is only one cause of unnecessary rework and poor coordination. In practice, organisational and human behaviour can also have significant impact on how design processes unfold. For instance, behaviour may change as time pressure increases. Such issues are not represented in the model.

6.3.6 Summary

All these factors were not explicitly considered in the model and could be investigated as part of future work. However, they are all essentially refinements to the basic scheme of resource-limited progressive iteration, occasionally set back by rework and change propagation. They are thus not expected to affect the main findings of the simulation study.

6.4 Limitations and opportunities for further work regarding the simulation study

6.4.1 Limitations regarding DSMs used in the study

Most of the DSMs used in the simulation study are binary; thus, the results are determined mostly by the topology of the networks under study, and not their impact and likelihood values. Further work could focus on using more DSMs incorporating distinct impacts and likelihoods of change propagation between components.

Additionally, the size of the DSMs used is relatively small, and thus, the findings cannot be generalised to very large models. Further research should investigate whether the results of this study would apply to product decompositions of larger size. Nevertheless, the DSMs were sourced from existing engineering design research literature; despite their small size they are therefore appropriate to the research context.

6.4.2 Limitations regarding priority policies chosen for the study

The priority policies studied are relatively simple. They do not explicitly consider issues arising from concurrency such as occupancy of resources and slack. To reduce the lead time in highly concurrent processes, special attention should be paid to avoid idle times and task interruptions due to changes and rework. Another strategy might be to define rules that aim to ensure a reprioritised project includes a degree of robustness to future changes. Such issues are beyond the scope of the present article, but the model could potentially be extended to study them in future work. We also did not test priority rules that combine in-degree and out-degree measures. This is an important opportunity for further work given that Braha and Bar-Yam (2007) show that both in-degree and out-degree contribute to PD dynamics.

Another opportunity for further research is to study the application of prioritisation policies in initial project planning, complementing their use in response to unplanned events as studied in this article, and validating the simulation results against real-world projects.

7 Concluding remarks

Scheduling and prioritising tasks can have a significant effect on product development cost and lead time. In this article, the effects of different priority policies were evaluated using a new simulation model incorporating the combined effects of progressive iteration, rework and change propagation during design of interconnected subsystems. The model integrates these factors into one model which uses a system architecture DSM as the basis for design process simulation. The underpinning of the model is a stepwise progression of individual component's maturity levels during the design process. Components cannot be developed independently because interfaces between them constrain the activity sequence.

Task prioritisation decisions have to be made throughout the design process and, because of the certainty of unforeseen events, cannot be fully planned in advance. The prioritisation policies evaluated in this article can provide rules of thumb for practitioners when many jobs compete for attention and a quick decision is required with limited information. The simulations suggest that certain policies are effective in reducing unnecessary rework, regardless of the scope of the project and a number of other contextual variables. Equally important, some policies that seem reasonable, such as focusing effort on components whose progress is lagging behind the rest, were found to amplify the total amount of rework. Although the simulation may initially appear complicated, the policies that were evaluated are straightforward to apply, and thus, the findings should provide pragmatic insight.

Acknowledgments This work was supported by an Industrial CASE studentship funded by the UK Engineering and Physical Sciences Research Council and BT [EP/K504282/1]. Figures 2 and 3 were generated using Cambridge Advanced Modeller (CAM) (Wynn et al. 2010).

References

Abdelsalam HME, Bao HP (2006) A simulation-based optimization framework for product development cycle time reduction. IEEE Trans Eng Manag 53(1):69–85

Antonsson EK, Otto KN (1995) Imprecision in engineering design. ASME J Mech Des 117(2):25–32

Anzanello MJ, Fogliatto FS (2011) Learning curve models and applications: Literature review and research directions. Int J Ind Ergonom 41(5):573–583

Braha D, Bar-Yam Y (2004a) Information flow structure in large-scale product development organizational networks. J Inf Technol 19(4):244–253

Braha D, Bar-Yam Y (2004b) Topology of large-scale engineering problem-solving networks. Phys Rev E 69(016):113

Braha D, Bar-Yam Y (2007) The statistical mechanics of complex product development: empirical and analytical results. Manag Sci 53(7):1127–1145

Braha D, Maimon O (1997) The design process: properties, paradigms, and structure. Syst Man Cybern A Syst Hum IEEE Trans 27(2):146–166

Browning TR, Eppinger S (2002) Modeling impacts of process architecture on cost and schedule risk in product development. IEEE Trans Eng Manag 49(4):428–442

Browning TR, Yassine AA (2010) Resource-constrained multi-project scheduling: priority rule performance revisited. Int J Prod Econ 126(2):212–228

Brucker P, Drexl A, Mohring R, Neumann K, Pesch E (1999) Resource-constrained project scheduling: notation, classification, models, and methods. Eur J Oper Res 112(1):3–41

Buddhakulsomsiri J, Kim DS (2007) Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. Eur J Oper Res 178(2):374–390

Chang AST (2002) Reasons for cost and schedule increase for engineering design projects. J Manag Eng 18(1):29–36

Cho SH, Eppinger S (2005) A simulation-based process model for managing complex design projects. IEEE Trans Eng Manag 52(3):316–328

Christiansen TR (1993) Modeling efficiency and effectiveness of coordination in engineering design teams. PhD thesis, Department of Civil Engineering, Stanford University

Chtourou H, Haouari M (2008) A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. Comput Ind Eng 55(1):183–194

Clarkson PJ, Simons C, Eckert C (2004) Predicting change propagation in complex design. J Mech Des 126(5):788–797

Cohen GP (1992) The virtual design team: an object-oriented model of information sharing in project teams. PhD thesis, Department of Civil Engineering, Stanford University

Eckert C, Clarkson PJ, Zanker W (2004) Change and customisation in complex engineering domains. Res Eng Des 15(1):1–21

Einögg F (2009) Netzwerk-FMEA: Methodik und Anwendung. Unveröffentlichte Semesterarbeit, TU Munich, Institute of Product Development

Eppinger SD, Whitney DE, Smith RP, Gebala DA (1994) A model-based method for organizing tasks in product development. Res Eng Des 6(1):1–13

Ford DN, Sterman JD (2003a) The liar's club: concealing rework in concurrent development. Concurr Eng Res Appl 11(3):211–219

Ford DN, Sterman JD (2003b) Overcoming the 90% syndrome: iteration management in concurrent development projects. Concurr Eng 11(3):177–186

Garcia R (2005) Uses of agent-based modeling in innovation/new product development research. J Prod Innov Manag 22(5): 380–398

Grebici K, Goh YM, Zha S, Blanco E, McMahon C (2007) Information maturity approach for the handling of uncertainty within a collaborative design team. In: Proceedings of the 2007 11th International Conference on Computer Supported Cooperative Work in Design, Vols 1 and 2, pp 280–285.

Hartmann S, Briskorn D (2010) A survey of variants and extensions of the resource-constrained project scheduling problem. Eur J Oper Res 207(1):1–14

Hölttä-Otto K, de Weck O (2007) Degree of modularity in engineering systems and products with technical and business constraints. Concurr Eng 15(2):113–126

Huberman BA, Wilkinson DM (2005) Performance variability and project dynamics. Comput Math Organ Theory 11(4):307–332

Jarratt T, Eckert C, Caldwell N, Clarkson P (2011) Engineering change: an overview and perspective on the literature. Res Eng Des 22(2):103–124

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

Karniel A, Reich Y (2009) From DSM-based planning to design process simulation: a review of process scheme logic verification issues. IEEE Trans Eng Manag 56(4):636–649

Koh E, Caldwell N, Clarkson P (2012) A method to assess the effects of engineering change propagation. Res Eng Des 23(4): 329–351

Kolisch R (1996) Efficient priority rules for the resource-constrained project scheduling problem. J Oper Manag 14(3):179–192

Krishnan V, Eppinger SD, Whitney DE (1997) Model-based framework product to overlap development activities. Manag Sci 43(4):437–451

Lévárdy V, Browning TR (2009) An adaptive process model to support product development project management. IEEE Trans Eng Manag 56(4):600–620

Levitt RE, Thomsen J, Christiansen TR, Kunz JC, Jin Y, Nass C (1999) Simulating project work processes and organizations: toward a micro-contingency theory of organizational design. Manag Sci 45(11):1479–1495

Licht T, Schmidt L, Schlick CM, Dohmen L (2007) Person-centred simulation of product development processes. Int J Simul Proc Model 3(4):204–218

Lindemann U, Reichwald R (1998) Integriertes Änderungsmanagement. Springer, Berlin

Lindemann U, Maurer M, Braun T (2009) Structural complexity management: an approach for the field of product development. Springer, Berlin

Loch CH, Terwiesch C (1998) Communication and uncertainty in concurrent engineering. Manag Sci 44(8):1032–1048

Maurer M (2011) Komplexitätsmanagement für die industrielle praxis Unterlagen zur Vorlesung. Institute of Product Development, TU Munich

Newman M (2003) The structure and function of complex networks. SIAM Rev 45(2):167–256

O'Brien C, Smith SJE (1995) Design maturity assessment for concurrent engineering co-ordination. Int J Prod Econ 41(1–3): 311–320

Pasqual MC, de Weck OL (2012) Multilayer network model for analysis and management of change propagation. Res Eng Des 23(4):305–328

Repenning NP (2001) Understanding fire fighting in new product development. J Prod Innov Manag 18(5):285–300

Schlick CM, Duckwitz S, Schneider S (2013) Project dynamics and emergent complexity. Comput Math Organ Theory 19(4):480–515

Schmitz S, Wynn D, Biedermann W, Clarkson PJ, Lindemann U (2011) Improving data quality in DSM modelling: A structural comparison approach. In: Proceedings of the 18th International Conference on Engineering Design (ICED11)

Sered Y, Reich Y (2006) Standardization and modularization driven by minimizing overall process effort. Comput Aided Des 38(5):405–416

Smith RP, Eppinger SD (1997a) Identifying controlling features of engineering design iteration. Manag Sci 43(3):276–293

Smith RP, Eppinger SD (1997b) A predictive model of sequential iteration in engineering design. Manag Sci 43(8):1104–1120

Smith RP, Morrow JA (1999) Product development process modeling. Des Stud 20(3):237–261

Steward DV (1981) The design structure-system: a method for managing the design of complex-systems. IEEE Trans Eng Manag 28(3):71–74

Wynn DC, Eckert CM, Clarkson PJ (2007) Modelling iteration in engineering design. In: Proceedings of the 16th International Conference on Engineering Design (ICED07), pp 693–694

Wynn DC, Wyatt DF, Nair S, Clarkson PJ (2010) An introduction to the Cambridge Advanced Modeller. In: Proceedings of the 16th International Conference on Modelling and Management of Engineering Processes (MMEP 2010)

Wynn DC, Grebici K, Clarkson PJ (2011) Modelling the evolution of uncertainty levels during design. Int J Interact Des Manuf (IJIDeM) 5(3):187–202

Yang F, Duan GJ (2012) Developing a parameter linkage-based method for searching change propagation paths. Res Eng Des 23(4):353–372

Yassine AA (2007) Investigating product development process reliability and robustness using simulation. J Eng Des 18(6): 545–561

Yassine AA, Braha D (2003) Complex concurrent engineering and the design structure matrix method. Concurr Eng 11(3):165–176

Yassine AA, Whitney DE, Zambito T (2001) Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM). In: Proceedings of DETC 2001

Yassine AA, Joglekar N, Braha D, Eppinger SD, Whitney D (2003) Information hiding in product development: the design churn effect. Res Eng Des 14(3):145–161

Zhang X, Li Y, Zhang S, Schlick CM (2013) Modelling and simulation of the task scheduling behavior in collaborative product development process. Integr Comput Aided Eng 20(1):31–44