Res Eng Design (2018) 29:161–202
https://doi.org/10.1007/s00163-017-0262-7

ORIGINAL PAPER

Process models in design and development

David C. Wynn[1] · **P. John Clarkson**[2]

Received: 4 October 2016 / Revised: 25 June 2017 / Accepted: 30 June 2017 / Published online: 12 July 2017

Abstract Many models of the design and development process have been published over the years, representing it for different purposes and from different points of view. This article contributes an organising framework that clarifies the topology of the literature on these models and thereby relates the main perspectives that have been developed. The main categories of model are introduced. Their contexts, advantages, and limitations are considered through discussion of selected examples. It is demonstrated that the framework integrates coverage of earlier reviews and as such provides a new perspective on the literature. Finally, key characteristics of design and development process models are discussed considering their applications in practice, and opportunities for further research are suggested. Overall, the article should aid researchers in positioning new models and new modelling approaches in relation to state-of-the-art. It may also be of interest to practitioners and educators seeking an overview of developments in this area.

Keywords Process models · Design and development · Literature review · Organising framework

✉ David C. Wynn
d.wynn@auckland.ac.nz

[1] Department of Mechanical Engineering, University of Auckland, 20 Symonds Street, Auckland 1010, New Zealand

[2] Department of Engineering, University of Cambridge, Trumpington Street, Cambridge CB2 1PZ, UK

1 Introduction

In comparison to many other processes, the design and development process (DDP) is especially challenging to navigate and manage. Researchers have developed numerous process models to understand, improve, and support the DDP considering its particular characteristics. However, the complexity is such that no single model can address all the issues. Furthermore, the many models that have been developed are diverse in focus and formulation. This article aims to summarise the current thinking in the area by providing an up-to-date overview of DDP models, and by developing an organising framework that positions them in relation to one another.

The models we consider are motivated by, and aim to address, certain characteristics of the DDP that distinguish it from many other processes. In particular, the DDP tends to involve significant elements of novelty, complexity, and iteration. The following paragraphs introduce these interrelated issues and outline how process models can help to address them, before moving on to discuss this article's contribution.

First, considering novelty, "design processes seek to do something novel, once, whereas many other business processes seek to do the same thing repetitively" (O'Donovan et al. 2005). In consequence, every DDP is unique and involves a degree of uncertainty (Eckert and Clarkson 2010). New activities are typically discovered during projects (Karniel and Reich 2011); the process sequence is unpredictable, because tasks are progressively concretised and adjusted as work proceeds (Albers and Braun 2011); and decisions must often be based on inadequate or preliminary information (Antonsson and Otto 1995; Pich et al. 2002). These issues may be observed on all levels of scale,

from designers working alone through to complex development programs.

Second, considering complexity, large-scale concurrent engineering in particular involves many tasks and individuals, a densely connected web of information flows, and many interdependent design issues that must be considered simultaneously (Eppinger et al. 1994). Feedback processes within a DDP are also significant drivers of dynamic complexity. For instance, bringing on new staff to handle a peak in workload may cause quality problems that require even more work to correct later on (Reichelt and Lyneis 1999). DDP complexity seems to be increasing overall, for instance due to continuing introduction of new design issues and technologies, and increasingly fragmented disciplinary specialisation (Maurer 2017).

Third, considering iteration, it is well recognised in the literature that design and development are iterative in nature (e.g., Dorst and Cross 2001; Yassine and Braha 2003). Iteration can have numerous roles in the DDP, including: iteration to progress the design; iteration to correct problems or implement changes; and iteration to enable coordination within a process, or between a process and its context (Wynn and Eckert 2017). Managing and exploiting iteration are critical to design and development on any scale, yet can be difficult in practice due to the many perspectives that are possible.

To summarise, these characteristics and related issues mean that companies and individual designers may not fully understand the processes by which they generate their designs (O'Donovan et al. 2005). In consequence, the DDP is difficult to execute and manage effectively. Cost and schedule overruns are common (Reichelt and Lyneis 1999). Because effective design and development is critical to many organisations' performance, this has motivated much research to better understand such processes and how they might be supported and improved.

Research has suggested that process models can help to address the challenges outlined above in several ways. For example, while large-scale design and development processes do involve novelty, they also involve routine sequences and structures that can be modelled (Browning et al. 2006). Consequently, a common view is that these processes "are systems and can be engineered", a task which can be facilitated by process models and process modelling (Browning and Ramasesh 2007). Process models may also help to align process participants and their mental models. They are, therefore, important enablers of coordination, defined by Malone and Crowston (1994) as the management of dependencies among activities. This becomes more important as complexity and innovation increase (Zhang et al. 2015). Process models depicting best practice may be useful "to rationalise creative work, to reduce the likelihood of forgetting something important, to permit design to be taught and transferred, to facilitate planning, and to improve communication between disciplines involved in design" (Gericke and Blessing 2011). Models can also help to generate and communicate conceptual insights into the DDP. This is useful to researchers and educators, may inform practitioners, and may inspire the development of pragmatic support.

Although process models can, therefore, be helpful in understanding and handling the special characteristics of the DDP, those same characteristics make its modelling difficult. Despite extensive work undertaken since the 1950s, no single descriptive model is agreed to provide a satisfactory account of the design and development process (Bahrami and Dagli 1993). Indeed, this is probably not achievable. Similarly, in terms of prescriptive models developed to support or improve the DDP, there is arguably still "no silver bullet approach to achieve process improvement" (Wynn and Clarkson 2005). This is again unsurprising considering the complexity of the topic and the many issues involved.

1.1 Contribution of this article

As noted above, DDP models fulfil a number of purposes for practitioners, researchers, and educators. However, the design and development process involves many interrelated issues, and each model of the DDP embodies a selective viewpoint on those issues. We therefore contend that state-of-the-art understanding of the DDP and of best practice is not embodied in any one model—but in the set of models and the relationships between them.

This article reviews the models and clarifies their relationships. First, we contribute an organising framework which shows how models of design and development processes can be positioned in relation to one another. Second, we contribute a review and integrating summary of key DDP models. We will show that although a number of researchers have previously surveyed such models, the earlier literature reviews each focus on only a subset of the categories that we identify here. By describing key models, integrating the coverage of earlier reviews, and providing pointers for further reading, it is anticipated that this article will be useful to researchers seeking to position their work as well as to practitioners and educators seeking an overview of the approaches that have been developed. Insights regarding the advantages, limitations, and applications of the individual models are also provided along with suggested areas for further research.

1.2 Scope of the literature review

The body of relevant literature is expansive and incorporates a broad range of perspectives. As with any work

based on review of a large and established research field, decisions were needed on what to include and how to organise it. In this case, the decisions were guided by the authors' previous research into complex design processes. This involved industry case studies, model and method development, literature study, and practitioner experience.

The following decisions were made regarding scope. First, because designing is intertwined with the other work that takes place in a development project, we contend that these processes should be understood together. Therefore, the scope includes both design processes and development processes. Second, for the purposes of this article, the term 'model' refers to any explicit representation of a perceived or envisaged DDP situation, or any approach specifically intended to express and/or analyse such representations. A model may be expressed graphically, mathematically, computationally, and/or in written form. Third, we focus on models pertinent to engineering design and development. Although related topics such as user-centered design and product-service systems design are not explicitly treated, a number of the models that we review are relevant to all design activity and thus may be of interest to researchers working on these topics. Fourth, the article only considers models that explicitly include design activity, excluding those that focus entirely on other processes within the design and development context, such as manufacturing. Fifth, it was decided to focus on explicit models of process and to not discuss in detail topics such as product models and parametric models, even though such models do have implications for the design and development process. Sixth, models focusing on specific design issues such as design for assembly are not emphasised, nor are models specific to particular companies. Finally, work on computational design and design optimisation processes is considered out-of-scope.

The article is an integrative overview in which an organising framework is explained and illustrated by discussion of selected key publications. Therefore, although the framework aims to provide comprehensive coverage of model categories and to indicate the relationships between them, the bibliography does not comprise a complete list of all model variants nor all relevant publications. Pointers to more exhaustive but more narrowly focused reviews are provided where such work is available. Finally, we note that many DDP models could be interpreted or applied in different ways, which can cause difficulties arriving at an unambiguous classification. In this article, we seek to keep our analysis as grounded as possible by restricting our attention to how each model is described in its key supporting publications, as listed in the bibliography.

1.3 Methodology

We began with the organising framework first published by Wynn and Clarkson (2005) and subsequently expanded by Wynn (2007). We sought to substantially improve comprehensiveness of the framework and to complement it with research insight developed since these earlier publications. Identification of models to include in the updated framework began with study of earlier literature reviews (see Sect. 6.6). Original sources mentioned in these reviews were considered, and research journals were also consulted to find additional recent publications. Bibliographies and Internet search were used to progressively identify further relevant sources. This yielded a large number of models which were filtered according to the criteria of Sect. 1.2.

The framework described by Wynn (2007) was expanded and iteratively revised as relevant literature was digested. Our main consideration was to find a way to conceptualise, articulate, and visualise the relationships between diverse models while also allowing a relatively linear exposition. This led to a new framework having substantially different form and significantly expanded coverage to the original.

2 Organising framework

The framework was designed to cluster similar models together, such that models within each cluster can be meaningfully compared and such that the clusters themselves can be meaningfully related. To approach this, we note that each model is a simplification or abstraction of a perceived or envisaged situation, in which the form of the model is influenced by the intentions of the modeller (e.g., Pidd 1999; Browning et al. 2006; Maier et al. 2017). It follows that models can be meaningfully grouped according to (1) the characteristics of the targeted situation and (2) the overall purpose of the model. The organising framework that was developed from this concept comprises two dimensions each subdivided into several categories. These are introduced in the next subsections prior to discussing the models themselves.

2.1 Model scope dimension

The first dimension of the framework considers the scope, i.e., breadth of coverage of a model. This dimension is important because the framework organises models that range from an individual's mental activities during design through to complex development programs that may involve thousands of participants and multiple tiers of suppliers. These situations have quite different characteristics, which are reflected in the models.

Considering the relationship between these situations, Hall (1962) proposed a two-dimensional perspective of development projects in which the stage-based structure of a project's lifecycle is orthogonal to an iterative problem-solving process that occurs within each stage. Asimow (1962) similarly described the essentially linear, chronological structure of a project as its morphological dimension, and the highly cyclical, iterative activities characteristic of designers' day-to-day activities as the problem-solving dimension. Blessing (1994) refers to models concerned with Asimow's morphological and problem-solving dimensions as stage-based and activity-based, respectively. She also notes the existence of combined models which prescribe well-structured, iterative activities within each stage (e.g., Hubka 1982). Other models such as the Task DSM (Eppinger et al. 1994) represent individual tasks and their interrelationships. Their focus is in between the iterative problem-solving process and the overall structure of the DDP.

Combining these ideas, the model scope dimension of the framework comprises three categories:

- *Micro-level models* focus on individual process steps and their immediate contexts.
- *Meso-level models* focus on end-to-end flows of tasks as the design is progressed.
- *Macro-level models* focus on project structures and/or the design process in context. This can include the overall form of a project or program, organisational and managerial issues relating to a DDP situation, and/or the interaction between the DDP and the context into which a design is delivered.

2.2 Model type dimension

While the scope dimension groups models that target similar situations, the type dimension groups models that have similar overall purpose. Considering the literature and extending the classification of Wynn (2007), four model types were identified:

- *Procedural models* convey best practices intended to guide real-world situations.
- *Analytical models* provide situation-specific insight, improvement, and/or support which is based on representing the details of a particular DDP instance.
- *Abstract models* convey theories and conceptual insights concerning the DDP. Such models have yielded important insights into design and development, and have inspired the creation of pragmatic approaches, but many of them do not directly offer guidance for practitioners.
- *Management science/operations research (MS/OR) models* use mathematical or computational analysis of representative or synthetic cases to develop generally applicable insights into DDP issues.

2.3 Summary

The dimensions and categories of the organising framework are summarised in Table 1. Most models are possible to align against a single category within each dimension, but the categories are not mutually exclusive. Some models and publications contribute to several categories in a dimension. Some could arguably be categorised in different ways depending on how they are interpreted and applied.

Although the need for interpretation when categorising models cannot be eliminated entirely, the premise of this article is that any design or development process model can be assigned to at least one category within each of the framework dimensions. We contend that doing this can help to express a model's characteristics. Considering the two dimensions together allows the DDP models and the perspectives they represent to be clustered, and clarifies the context in which each model should be considered. To illustrate, selected key models are positioned against the framework in Fig. 1.

The next sections elaborate the framework by discussing selected models in each category and some of the main ideas embedded in them. Discussion is organised primarily around the scope dimension and secondarily around the type dimension, thereby spiralling outwards through layers of the framework as depicted in Fig. 2. This was found suitable to establish a linear presentation of the issues. Sections 3, 4, 5 discuss models on the micro-, meso-, and macro-levels, respectively. Section 6 draws on the framework to consider the implications of key DDP characteristics on models and modelling. Section 6 also discusses how DDP models are used in engineering practice and suggests areas for future research.

3 Micro-level models

To recap, models of the DDP on the micro-level focus on individual process steps and their immediate contexts. Such models typically emphasise individual or small group situations. The next subsections describe micro-level procedural, analytical, abstract, and MS/OR models in turn.

Table 1 The organising framework comprises two dimensions, each with several categories

Dimension	Category	Models in this category
Scope	Micro-level	Focus on individual process steps and their immediate contexts
	Meso-level	Focus on end-to-end flows of tasks as the design is progressed
	Macro-level	Focus on project structures and/or the design process in context
Type	Procedural	Convey recommendations of best practice
	Analytical	Provide ways to model specific situations for analysis/improvement/support
	Abstract	Convey theories and conceptual insights into the DDP
	MS/OR	Develop insights by mathematical/computational analysis of representative cases

Fig. 1 Positioning key models of design and development within the framework generates a map of the literature

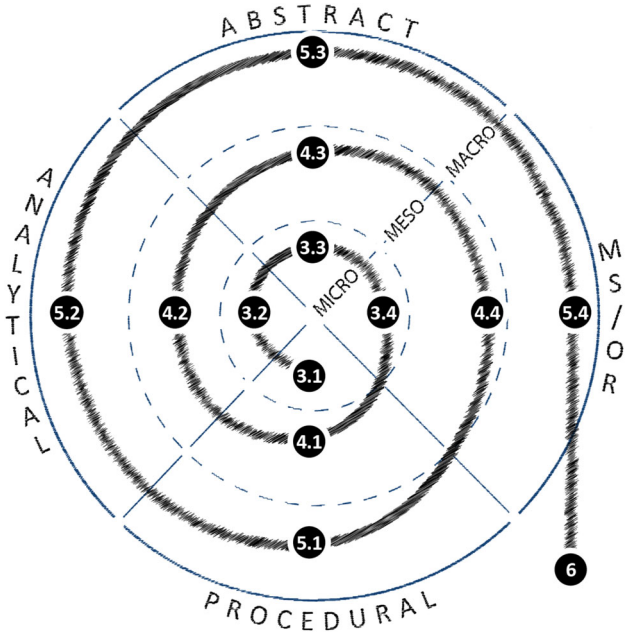

Fig. 2 The narrative in this article spirals outwards through the layers of the framework. *Numbers* refer to subsections where corresponding categories are discussed

3.1 Micro-level procedural models

Micro-level procedural models provide prescriptive guidelines for the design and problem-solving activity which occurs at many points throughout a project. There are four main groups of model in this category.

First, on the most conceptual level, certain overall strategies for design are recommended by many authors. Foremost is the idea that designers should proceed systematically and resist their preconceptions. Evbuomwan et al. (1996) review the early work incorporating this recommendation, including Marples (1961), Jones (1963) and Archer (1965). They find that all these authors prescribe the three main steps of analysis, synthesis, and evaluation. Analysis involves focusing on a problem and structuring it into a set of objectives. Synthesis involves generation of a range of candidate solutions. Evaluation involves the critical appraisal of those solutions against the objectives, to rationally select between them and/or to drive iterative improvement. Models incorporating this strategy have often been described as problem oriented (Wynn and Clarkson 2005). They are based on the premise that designers can formulate solution-neutral problem statements, and that doing so is useful to direct their creative insights with systematic reasoning. Another common recommendation is to begin by deliberately expanding the perceived boundaries of a problem, e.g., by relaxing constraints, attempting to reframe the problem, or attempting to perceive it on a higher level of abstraction. This may help to avoid artificial overconstraint and ensure that a broad range of potential solutions is considered. A third common strategy is to decompose a problem into simpler subproblems with well-defined interactions as early as possible, such that the subproblems can be addressed individually prior to recombining solutions (Fig. 3). This is thought to encourage "the discipline to proceed systematically" and enable "rationally organised division of labour" (VDI2221 1987). Overall, the design strategies discussed in this paragraph are desirable but might not provide much practical guidance for implementation. For this reason, they are often embedded in more concrete procedural models, many of which are reviewed elsewhere in this article.

The second group of models in this category comprises more concrete systematic methods that support the execution of specific design steps. Examples include approaches to promote creativity such as C-Sketch (Shah et al. 2001); use of morphological matrices to search for possible combinations of working principles (Pahl et al. 2007); and decision methods such as the analytic hierarchy process (AHP) (Saaty 1987) and the controlled convergence method (Pugh 1991). Axiomatic Design recommends a process of zig-zagging through a hierarchy of functional requirements (FRs) and design parameters (DPs) while striving to satisfy design rules derived from two axioms—presented as "fundamental truths" about the characteristics

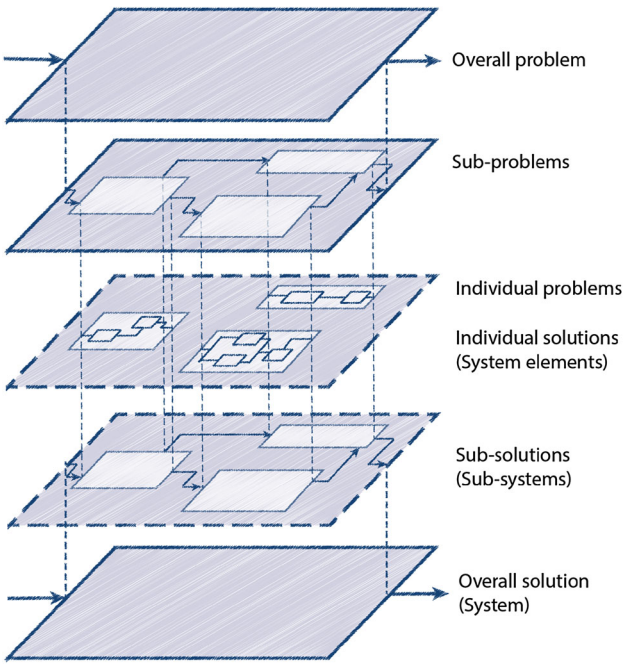

Fig. 3 Method of structuring problems and systems (VDI2221 1987). Reproduced with permission of the Verein Deutscher Ingenieure e. V

of good designs (Suh 1990). TRIZ/ARIZ offers methods to support innovation by identifying and resolving contradictions in a design situation (Altshuller 1999).

Such systematic methods (and there are many more) may be useful at many points in a DDP. Pugh (1991) describes them as "the designer's tool-kit" which allows discipline-specific engineering knowledge to be applied efficiently and effectively. Hubka (1982) expresses a commonly held view by recommending systematic procedures when searching for concepts to cover a wider search space. He also suggests that a systematic approach can be particularly beneficial in all review and revision activities. Archer (1965) proposes that systematic approaches are particularly useful under one or more of three conditions: when mistakes can have significant consequences; when the likelihood of making mistakes is high, for example due to inexperience; and/or when the situation is complex, characterised by many interacting variables.

A third group of models recommend procedures for solving problems encountered during a DDP. The archetypical procedural model of problem-solving is the Shewhart Plan–Do–Check–Act (PDCA) cycle, which dates from 1939 (Moen and Norman 2010). The PDCA cycle recommends an iterative process in which thorough up-front analysis of the problem (Plan) and solution implementation (Do) are followed by seeking feedback (Check) and adjustment of the solution (Act). This iterative feedback process may help to obtain robust and validated solutions. It is thought to be especially useful where the problems being solved are ill-defined, complex enough that they cannot be easily grasped, are set in a changing context, and/or in a context where the solution can influence the nature of the problem (Wynn and Eckert 2017). Related to PDCA, more recent problem-solving models such as Define–Model–Analyse–Improve–Control (DMAIC), Look–Ask–Model–Discuss–Act (LAMDA), A3 Problem-Solving, and Kepner–Tregoe methodology also often appear in DDP practice, and include similar iterative elements. For further discussion and review of prescriptive problem-solving models in the DDP context, the reader is referred to Mohd Saad et al. (2013).

The fourth and final group of micro-level procedural models concern negotiation protocols for design. The issue addressed by these models is that different stakeholders in a design problem are usually responsible for different variables and objectives, some of which will be in conflict (Klein 1993). Negotiation protocols or methodologies prescribe processes for interaction between human and/or computational stakeholders, to assist them in reaching a mutually satisfactory outcome without excessive iteration (e.g., Lewis and Mistree 1998; Jin and Geslin 2009).

3.2 Micro-level analytical models

Micro-level analytical models provide formalisms to assist in the modelling of design knowledge from a process perspective. They represent individual process steps and decisions, indicating how they relate to specific features of the design context. The contextual information is thought to provide "guidance to reapply knowledge at the most appropriate time" (Baxter et al. 2007).

An early approach called PROSUS uses a matrix system for knowledge modelling during the design process (Blessing 1994). The matrix columns are defined by three micro-level activities, namely generate, evaluate, and select. The rows denote the problem, requirements, functions, concept, and the detail design. As the designer proceeds through iterative cycles, they are intended to capture their knowledge regarding proposals, arguments, and decisions within the appropriate cells of a PROSUS matrix. It is proposed that a different matrix should be used for each design situation encountered. A subsequent approach called the design history system (DHS) represents technical knowledge relevant to a design in terms of the processes and decisions that generated it (Shah et al. 1996). DHS represents: design steps; product data such as assembly relations and geometry, including successive versions and configurations; the relationships between design steps and product data they operate on; and the rationale underlying decisions. Emphasis is placed on intelligent querying of the history to help designers understand and reuse past designs. Addressing similar issues, the engineering history base (EHB) of Taura and Kubota (1999) allows designers to model the rationale behind design attributes in two ways: their relationships to design goals; and the need to work within constraints created by the previous decision-making activities. A prototype software tool allows the reasoning behind a particular attribute to be traced through the process. Both these papers focus on defining classes and relations to structure knowledge databases, and propose form-oriented interfaces. The Decision Rationale Editor (DREd) 2.0 tool reported by Aurisicchio and Bracewell (2013) instead uses a less formal graphical network representation building on the gIBIS approach, which allows designers to model the rationale structure supporting each process step or decision. Deployment and acceptance in an industry context were achieved (Aurisicchio and Bracewell 2013).

Other models focus on representing micro-level process knowledge with the specific objective of guiding a designer from one step to the next. For example, Signposting was developed to support rotor blade design by guiding the designer towards tasks that are available and appropriate for each design context that is reached (Clarkson and Hamilton 2000). The unique feature of this approach is the

notion that designer confidence is an important factor driving task selection. In Signposting, a task is considered available if the designer indicates that their confidence in its input parameters meets specified thresholds, and appropriate if completing the task is expected to increase confidence in one or more parameters. In this context, high confidence in a parameter means that its value is detailed, accurate, robust, well understood, and physically realistic (Clarkson and Hamilton 2000). In a prototype implementation, the designer indicates their confidence in each design parameter, and the tool proposes tasks that are available and appropriate to attempt next. The manufacturing integration and design automation system (MIDAS) also aims to dynamically guide the designer through the design process, in this case using a hierarchical grammar-based model (Chung et al. 2002). In MIDAS, a design process is initially represented as a flow of logical tasks including inputs and outputs. This expresses what needs to be done on a high level of abstraction. As the process is executed, a database of production rules is consulted to detail logical tasks as they are encountered, replacing them on-the-fly with more detailed process flows. These can comprise more concrete logical tasks and/or atomic tasks, which encapsulate a specific approach for completing a step. Each production rule represents a possible strategy for approaching the logical task that it replaces. MIDAS includes a way to roll back the process instantiation and prompt the designer to try another strategy, which is needed when design data produced by a task do not satisfy constraints.

Finally, a third group of analytical models on the micro-level concern coordination support. For example, the agent-based decision network (ADN) of Danesh and Jin (2001) manages the process of decision-making and negotiation of solutions among agents, embedding models of a design problem alongside normative procedural models of the negotiation process, such as those mentioned in the previous subsection.

3.3 Micro-level abstract models

To recap, abstract models of the DDP focus on presenting insights about the process without prescribing how it should be approached. On the micro-level, such models concern the forms of reasoning, the elementary activities, and/or the types, structures, and evolutions of knowledge that occur during design. Insights from such work are essentially domain-independent.

The foci of these models may be illustrated by considering the categorisation of design situations discussed by Gero (1990, 2000). In routine designing, "all the necessary knowledge is available" (Gero 2000). Routine design problems can be seen as search problems and in principle can be solved using the conventional algorithms (Maher 2000). Nonroutine designing, in contrast, is thought to be more difficult to automate. Gero argues that nonroutine situations can be further divided into two subcategories. First, in innovative designing, "the context that constrains the available ranges for the variables is jettisoned, so that unexpected values become possible" (Gero 2000). Second, in creative designing, new variables may be introduced during the design process allowing truly novel designs to be produced (Gero 2000).

Models in this subsection focus mainly on nonroutine design processes including both subcategories. Researchers have identified important characteristics of such processes that are reflected in their models. These include:

- *Designing starts with ill-defined problems* Design problem specifications are often incomplete, inconsistent, and/or vague, because people do not fully understand the context, constraints, and possibilities before design begins. One factor separating nonroutine design from routine situations is that stakeholder needs may or must be interpreted, reformulated, renegotiated, and concretised (Smithers 1998).
- *Design problems and solutions coevolve* Considering possible solutions highlights new aspects of ill-defined problems and may lead to them being reframed. This may change the constraints on possible solutions and may change what is considered to be a good solution (Dorst and Cross 2001).
- *Designing is partly solution-oriented* Empirical research has indicated that designers prestructure problems to solve them. That is, existing knowledge and previous experiences are influential in the solution process (Hillier et al. 1972). Models taking this view are often called solution-oriented (Wynn and Clarkson 2005). According to Kruger and Cross (2006), they are usually considered to be more realistic representations of the designer's thought process than models which suggest the top–down and abstract-to-concrete strategy exemplified in Fig. 3.
- *Designing creates new parameters and generates new knowledge* Whereas routine processes involve finding suitable values for parameters whose existence is known, nonroutine designing involves modifying constraints and/or introducing new variables that were not originally anticipated (Gero 2000). New knowledge relevant to the design process is also generated as design proceeds.
- *Designing involves hierarchical structures* Solving a design problem often generates new problems at a more detailed level. Problems lower in the hierarchy are defined and constrained by partial solutions higher up (Guindon 1990).

- *Designing is situated* Each step in the design process influences the design situation, including the designer's knowledge, which in turn influences and constrains future design activity (Gero and Kannengiesser 2004).
- *Designing is progressive and iterative* As indicated above, a design solution is not generated in a single step but is approached progressively and iteratively. There are several perspectives on what gets revisited during micro-level iterations, and why (Wynn and Eckert 2017).

The next subsections discuss selected process models that each emphasise some of these characteristics of nonroutine design. The models convey insights that might be useful for teaching design, as well for developing AI approaches to either assist or automate design reasoning (Ullman et al. 1988). They might not all directly support working designers, but models discussed in this subsection have explanatory power and some have inspired the development of procedural and analytical work.

3.3.1 *Models that represent design as logical or formal operations*

The first group of abstract micro-level models represent designing in terms of formal or logical operations. These models are developed mainly from theoretical considerations regarding the properties of design problems and the design process. Motivations for such work include that if the logic of designing could be understood and specified formally, insights might be systematically derived and aspects of the process might be supported or automated with suitable reasoning algorithms.

In one seminal paper of this type, March (1976) developed the Production–Deduction–Induction (PDI) model which clarifies how creative, evaluative, and learning processes operate and interact when designing. The model comprises three phases that repeat in an iterative cycle. In the first phase, the designer considers a desired situation in view of their existing knowledge to speculate a possible design solution. This is seen as productive or abductive reasoning. In the second phase, the candidate solution's behaviour is predicted considering its form and relevant physical principles. This is deductive reasoning. In the third phase, new knowledge concerning probable general relations between solutions and their behaviours is induced from the specific case just analysed. The cycle then repeats with the benefit of this new knowledge. While deductive reasoning is analytic, abductive reasoning and inductive reasoning are synthetic. That is, their results are influenced by the context, including the knowledge and experience of the designer.

General design theory or GDT (Yoshikawa 1981) aims to define a formal logic of design. Here, in keeping with the scope of the present article, we do not discuss the formalism but focus on the process models associated with GDT. First, the evolutionary design process model (EDPM) focuses on how designers work with multiple representations of an emerging design (Tomiyama et al. 1989). According to the EDPM, design proceeds by progressively extending a metamodel from which the different product models can be derived. On each of a series of cycles, a problem is identified, specific model(s) are derived from the metamodel to analyse the design, allowing the problem to be resolved and leading to information being added to the metamodel. This is said to continue until a fully detailed design is reached. Tomiyama et al. (1989) argue that this is a mainly deductive process, complemented with additional logic operations to handle the multiple parallel paths considered during design and the need for backtracking when a problem is reached that cannot be solved by deduction. Second, Takeda et al. (1990) extend this work, placing greater emphasis on how the design process is directed from one step to the next and on the forms of logic involved. Their extended EDPM involves two levels. On the object level, the designer first develops a solution suggestion from awareness of a design (sub)problem, and then develops, details, and evaluates their proposed solution. On the action level, they decide on next steps if evaluation reveals contradictions in the proposal. Takeda et al. (1990) argue that suggesting a solution from awareness of a problem is achieved by abduction; developing details of the solution and evaluating it are both deduction; and causes of identified contradictions are found through a form of logic called circumscription. In their model, the causes of contradiction constitute new variables and a new problem to be addressed in a future design cycle. Third, Tomiyama (1994) devise a further improvement, called the refinement model, in which design is seen as a process to complete the specifications as well as to define design attributes. A detailed analysis and critique of GDT is provided by Reich (1995). Focusing mainly on the formal axioms and theorems rather than the process models, Reich (1995) concludes that the approach "cannot be an adequate description of real design", although, he argues, it might still provide useful "guidelines" for CAD system development.

Zeng and Cheng (1991) also take a formal approach. They focus on how reasoning at each step is situated in the outcome of previous design cycles, developing a recursive logic scheme to represent this process. Zeng (2002) integrates these ideas into his axiomatic theory of design modelling. This formally presents designing as a cycle of synthesis and evaluation which operates on a hierarchical structure defining the evolving design and its environment.

On each cycle, the synthesis of partial solutions contributes to the evaluation criteria for future cycles.

Braha and Reich (2003) build on the formal design theory (FDT) of (Braha and Maimon 1998a) to develop the coupled design process (CDP) model. CDP provides a mathematical formalism which emphasises the role of exploration in progressing a design. In overview, designing is modelled as a repeating cycle of a closure operation followed by a selection operation. The closure operation, representing exploration, involves creating a set of design descriptions which do "not differ substantially" from the output of a previous design cycle. This is referred to as a closure set. The selection operation then focuses attention on one or more design descriptions from the closure set, which form seeds for the next cycle. In CDP, each design description comprises both specifications and solutions, which are elaborated together until the design is complete. Braha and Reich (2003) argue that their model allows concepts from the mathematics of closures to be interpreted to provide insights into design, and furthermore argue that GDT is a special case of CDP. On the other hand, unlike GDT, Braha and Reich (2003) do not discuss how their formalism might be implemented computationally.

The final model to be mentioned in this subsection is the C-K theory introduced by Hatchuel and Weil (2003, 2009). These authors argue that the two issues of creativity and the expansion of knowledge are fundamental to understanding designing, but are not comprehensively integrated within earlier models. C-K theory aims to address this by presenting designing as a process of traversing back and forth between two structured and expanding spaces. Knowledge space K comprises statements representing the designer's knowledge. Concept space C comprises propositions relating to the emerging design concept(s). These are undecided in that they are not yet known to be true or false. Designing is conceptualised as a set of operations that are applied to expand the knowledge structures in conjunction with the concept space. It concludes when the propositions necessary for a design have been developed and found to be true. Several formalisms have been developed considering the ideas of C-K theory (e.g., Kazakçi 2009; Salustri 2014). Some support tools and industrial applications using the theory are discussed by Hatchuel et al. (2004). A 2014 review concluded that C-K theory has been developed, applied, and adopted in more than 100 publications, and that it provides a framework which may be able to integrate earlier theories of design (Agogué and Kazakçi 2014).

3.3.2 *Models that represent design as elementary abstract processes*

Some models decompose the design process into abstract steps independently of a mathematical formalism or analysis of inference types. One such model is the Function–Behaviour–Structure (FBS) framework (Gero 1990). This is based on the idea that all designs can be represented in terms of: functions, which describe what the design is for; behaviours, which describe what it does; and structures, which describe what it is (Gero and Kannengiesser 2014). FBS considers that designing occurs through eight transitions between these domains, defining the following processes: (1) formulating a problem, in which required functions are transformed into behaviours a design solution should exhibit; (2) solving the problem, through an iterative cycle in which desired behaviours are considered to create a structure representing the design, which is analysed to determine its actual behaviours, which are compared to the desired behaviours leading to design improvements (Gero 1990; 3) "focus shifts, lateral thinking, and emergent ideas" which arise while considering the design's structure (Gero and Kannengiesser 2014); and (4) documenting the solution. Gero and Kannengiesser (2004) extend this model to include the situated nature of design activity. They contend that design insights are generated not only from interactions within the designer's mind, as per item (3) above, but also by reinterpretation triggered when design ideas interact with the emerging design representation. To incorporate these ideas, Situated FBS decomposes designing into 20 transformation processes that transition among FBS domains in the external world, the designer's interpretation of it with respect to their emerging design, and the world they expect to produce with that design. These processes are shown in Fig. 4. Overall, Gero and Kannengiesser (2004) contend that their models differ from most others in "explicitly" representing the steps of reformulating the design and/or problem as new information is generated. Gero and Kannengeiser (2014) write that FBS and Situated FBS offer conceptual tools for understanding designing and provide bases for uniform coding of design protocols, allowing design activity to be studied independently of domain. The FBS framework and its underlying product model mentioned at the start of this paragraph have also been widely adopted to structure conceptual, computational, and empirical studies (e.g., Howard et al. 2008; Hamraz et al. 2013).

Other micro-level abstract models conceptualise the design process as an evolutionary system. For example, Hybs and Gero (1992) propose that the solutions considered during design can each be conceptualised as a genome, in which individual genes represent subsolutions. These authors develop a variant of FBS which illustrates that design can be viewed in terms of two genetic operators iteratively applied to a population of potential solutions: crossover, in which subsolutions are transplanted across designs, and mutation, in which subsolutions within one solution are changed by redesign. Maher and Poon (1996)

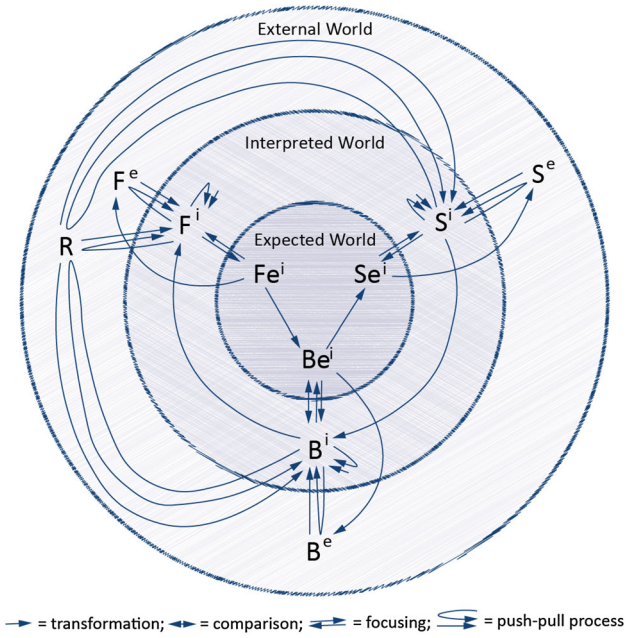

Fig. 4 Situated FBS views designing as a series of steps that are triggered by, and affect, emerging models of function (F), behaviour (B), structure (S), and requirements (R). The emerging models exist in three worlds and 20 types of step are possible, as shown. Key: X^e = external representation of X (where X is F, B, or S). X^i interpreted representation. Xe^i expected representation. Reproduced from Gero and Kannengiesser (2004) with permission of Elsevier

apply a similar evolutionary perspective with a focus on exploration in design. In this context, exploration refers to the process by which designers come to understand more about a problem as they consider potential solution concepts. In their model, Maher and Poon (1996) propose that problem and solution can be conceptualised as evolving genomes that influence the fitness function for each other. This is described as a coevolutionary process which proceeds until problem and solution definitions are both acceptable and compatible with one another (Maher 2000).

3.3.3 *Models that represent design as elementary operations*

A possible criticism of some models discussed above is their highly conceptual nature. This may cause difficulties interpreting them for application to real design problems. Although certain insights have been embedded in research prototypes, the objective of some authors to establish a mathematical basis for designing that allows its implementation in mainstream CAD does not seem to have been achieved yet.

Other authors approach the challenge of decomposing designing into elementary activity by focusing on more concrete operators and the specific knowledge structures or domains they operate on. For instance, Ullman et al. (1988) develop the Task–Episode–Accumulation (TEA) model to explain nonroutine mechanical design by analysing protocol recordings of designers working on such problems. Their model describes design as a series of tasks, each comprised from episodes that are undertaken to achieve goals. In turn, the episodes are decomposed into series of primitive operators falling into three categories: select, evaluate, and decide. The primitive operators are applied to the design state, which comprises all information about the emerging design. Key features of the TEA model include: design alternatives exist only within episodes, and as such, the design is incrementally reached through an accumulation of operators' results; the designer's working memory is explicitly modelled alongside operators to manage its limitations by loading and unloading relevant information; and goals are managed on a step-by-step basis, not in response to an overall plan. TEA, therefore, reflects observed designer behaviour in which an initial concept is "developed and gradually extended to accomplish the design goals" (Ullman et al. 1988). In common with other micro-level abstract models, this differs substantially from the systematic decomposition approaches exemplified by Fig. 3.

Finally, other models in this category were derived from the literature with a view to integrating key insights. For example, Chandrasekaran (1990) argues that AI approaches to design can be viewed as an iterative cycle of propose, critique, and modify. They review ways to approach each step. For example, the first step of solution proposal can be approached by algorithmic methods such as decomposition and recombination, constraint satisfaction, and so forth. A design process involves a mixture of approaches according to the characteristics of each subproblem encountered. Chandrasekaran (1990) argues that appropriate approaches can be selected dynamically by a controller which structures the task and chooses methods appropriate to each subgoal. Sim and Duffy (2003) develop a model of designing as a cycle of activity that is executed by a situated agent operating on input knowledge and producing output knowledge, in the context of individual objectives. They show that elementary activities described in the design literature can be categorised into three groups: design definition activities; design evaluation activities; and design management activities. Srinivasan and Chakrabarti (2010) also review elementary task models in the literature and argue that they can be mapped onto "a general problem finding and solving cycle" comprising the four activity types of generate, evaluate, modify, and select (GEMS). The outcomes of each activity are represented in terms of constructs that describe the emerging design and its operating principles. These constructs, namely State change, Action, Parts, Phenomenon, Input, oRgans, and

Effects (SAPPhIRE) were developed by Chakrabarti et al. (2005) based on the review and synthesis of earlier work.

3.3.4 Summary

Overall, micro-level abstract models highlight the iterative nature of designing and the need to respond to new information generated or revealed during that process. Such models provide theories and descriptions of design cognition, linking design activity to details of the emerging design and to knowledge and information about it. Most of these models and theories are based on first-principles reasoning and the supporting publications often do not emphasise an empirical basis or real-world validation. Nevertheless, some have received fairly wide interest and have been found to provide useful insight for real-world situations. According to Reich (1995), "each theory provides one perspective, broad, or limited, that may improve design understanding and practice." For further discussion of work in this category, the reader is referred to Eder and Weber (2006), Le Masson et al. (2013), and Chakrabarti and Blessing (2015).

3.4 Micro-level MS/OR models

To recap, the MS/OR category of our framework concerns models which apply mathematical or computer analysis to generate general insights from representative or synthetic situations. While many researchers have developed models of this type on the meso- and macro-levels (as described in forthcoming sections), we found relatively little on the micro-level once work on computational design and design optimisation is excluded. Some examples are given in the next paragraph.

First, Braha and Maimon (1998b) develop a mathematical model based on the principle that designing is characterised by progressive addition of attributes and relationships. Their model, based on an entropic perspective of design complexity, shows how progress causes an increase in the emerging design's complexity and consequently increases the effort required to progress it further. Second, Zeng and Yao (2009) use computer simulation to study the impact of different design strategies within their axiomatic theory of design modelling, which was discussed earlier. This theory suggests that different design solutions emerge through three levers: reformulating the problem; changing the designer's approach to the synthesis steps that occur on each design cycle; and altering the sequence of addressing problems that emerge while designing. Zeng and Yao (2009) implement their axiomatic model in an algorithm for generating a finite-element mesh—which they argue is representative of common design problems—and use simulated cases to show that adjusting these levers does indeed result in different solutions. Third, Kazakçi et al. (2010) develop a computer model to simulate designing according to the C-K theory principle that designs emerge through the interplay between concepts and knowledge. In this model, graph structures are used to represent the concept and knowledge spaces. These structures evolve through stepwise operations that reflect the steps of designing according to C-K theory. For example, one such operation involves generating a connection between two nodes in K space—this is simulated by selecting the nodes at random. Kazakçi et al. (2010) use their simulation to study how attention should be distributed between developing design concepts and undertaking research to develop relevant knowledge. They conclude that emphasising the former may generate a design solution more quickly, while the latter may help to ensure the solution is robust. Finally, another area of work that could contribute to this category is computational creativity, an emerging topic that aims to generate insights into creative activity by simulation of the processes involved (Sosa and Gero 2016). However, such models often focus on non-engineering domains and are thus outside the scope of this article.

To summarise, this is the least populated of the categories in our organising framework. Accordingly, there seems to be an opportunity for further research to apply mathematical and computational modelling to investigate the implications of micro-level models of engineering design activity, such as those discussed in Sect. 3.3.

4 Meso-level models

Having completed the discussion of micro-level models, we now move on to consider meso-level models. To recap, while micro-level models focus mainly on individual activities in their context, meso-level models concern end-to-end flows of activity that occur during design and development. Procedural, analytical, abstract, and MS/OR models on the meso-level are discussed in the next subsections.

4.1 Meso-level procedural models

Meso-level procedural models aim to support the effective generation of good designs by prescribing a systematic design process. A noteworthy early example was published by Evans (1959), who developed a spiral form to highlight the iterative nature of the design process (Fig. 5). Noting that one of the most fundamental characteristics of design is the need to find trade-offs between interdependent factors, Evans argues that design cannot be achieved by following a sequential process alone. He proposes that a

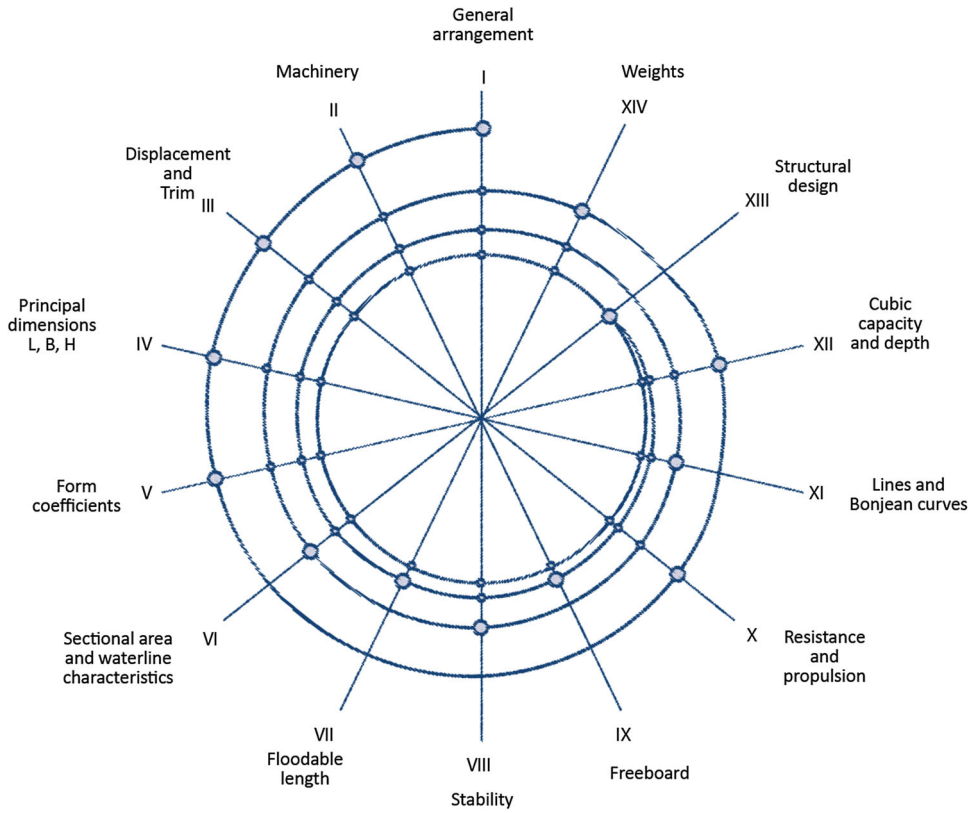

Fig. 5 Evans' meso-level model of the ship design process emphasises a structured cycle of convergence on key design objectives. Reproduced from Evans (1959). Reproduced with permission from the American Society of Naval Engineers

structured iterative procedure is adopted to resolve such problems; early estimates are made and repeatedly refined as the design progresses, until such time as the mutually dependent variables are in accord. As the project progresses, these design considerations are gradually refined by repeated attention in the indicated sequence until a balanced solution is reached. At each iteration, the margins available to absorb changes decrease as the interdependencies are gradually resolved, smaller modifications are required, and different methods may be applied to each problem. Evans notes that the effort required and the number of people that can be brought to bear increase as the solution converges.

Other models in this category present the design process as a series of stages, each of which further concretises the design by creating more information about it. This stage-based form is exemplified in the early work of French (1999), (Fig. 6), originally published in 1971. Later models focusing on mechanical design, notably in the work of Hubka and Eder (1996) and Pahl et al. (2007), prescribe detailed lists of working steps for each stage. These models define how to create the specific forms of information that constitute a mechanical design, progressing from abstract to concrete with the working steps organised such that each stage establishes objectives and constraints for the next. They depict "feedback" between the stages, which indicates the possibility of undesirable rework as well as inter-project and generational learning. Process models of this type are strongly influenced by models of the information structures that define a mechanical system design and its operation (e.g., Hubka's theory of technical systems and design processes, Hubka and Eder 1996, see also Sect. 4.3). Some years ago, Cross and Roozenburg (1992) argued that most had converged upon a consensus form, which is exemplified by Hubka's model (Fig. 7). More recently, some researchers have mapped numerous models onto proposed canonical stages to compare them (e.g., Howard et al. 2008; Gericke and Blessing 2012; Costa et al. 2015; Bobbe et al. 2016).

Overall, prescriptive stage-based models promote the idea that following a structured and systematic process will lead to a better result. For example, Pahl et al. (2007) state that following their steps ensures that nothing essential is overlooked, leading to more accurate scheduling and resulting in design solutions which may be more easily reused. Although (or because) they are popular, these models have also attracted critique. For example, the models emphasise original design cascading from stakeholder needs (Weber 2014), while real-world projects often place strong limitations on the early concept design, with constraints such as existing product platforms and legislative requirements often predetermining the form of the solution (Pugh 1991). Considering coverage of the models, Gericke and Blessing (2011) argue that although

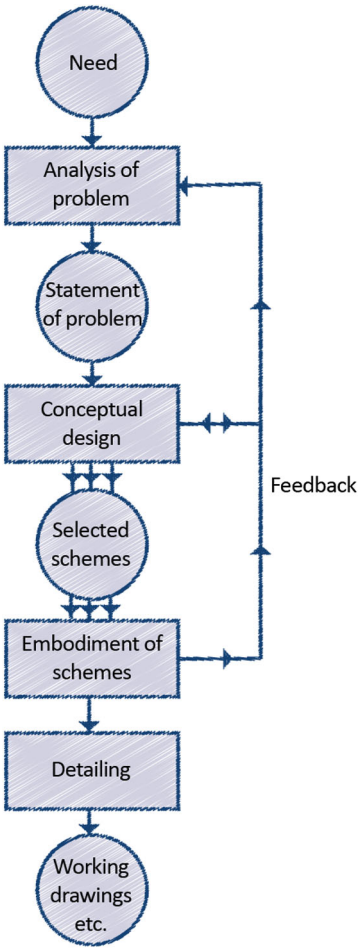

Fig. 6 Block diagram of design process. Figure and caption reproduced from French (1999) with permission of Springer

procedural models have been adapted to different disciplines, few integrate across them. Other researchers question the pragmatism of mandating a stage-based form. Whitney (2004), for instance, argues that the top–down ideal as represented in these models is clearly desirable, but practical considerations mean that this often merges with a bottom–up fitting-together of existing partial solutions. One reason for this discrepancy is that "a top–down process is very challenging intellectually", because it requires "imagining subassemblies and parts before they are known in any detail" (Whitney 2004). Konda et al. (1992) also point out that in collaborative design, participants use different analogies to represent the emerging design and must negotiate solutions, such that the idealised top–down approach proceeding from abstract to concrete may be difficult to maintain. Andreasen et al. (2015) summarise some of these concerns when writing that systematic approaches "only give a sparse insight into actual design, whilst giving the impression of rationality, which is not at all present".

Despite perceived limitations, the prescriptive meso-level model forms outlined here have been adapted and applied in many publications proposing discipline-specific design process models. For instance the general form of Evans' spiral model is still in use after more than five decades in fields from naval architecture (Rawson and Tupper 2001) to software engineering (Boehm 1988). Stage-based forms may be found in Dym et al. (2014), Ullman (2015), Pugh (1991), Roozenburg and Eekels (1995), the VDI guideline 2221 (VDI2221 1987), and many other publications.

4.2 Meso-level analytical models

The models described in the previous subsection recommend useful procedures and working steps for design. Although prominent in research and education, they are arguably not specific or detailed enough to guide many real-world situations. For example, in the design of complex products such as aircraft, Step 8 alone from Hubka and Eder (1996)'s model (see Fig. 7) typically involves some highly specialised working steps, spans several years, and involves hundreds or thousands of personnel and multiple tiers of suppliers. The meso-level analytical models discussed in this section should be better positioned to provide support in such contexts, because they are concerned with the specific steps that do or should occur within a company and/or design context. They help companies to portray specific DDPs as discrete tasks that interact through well-defined transfers of information to form an end-to-end flow. The premise is that modelling the detail of tasks and their organisation can support design, management, and improvement of meso-level processes.

One factor that delineates families of models within this category is how the relationships between tasks are treated (Wynn 2007). Our extended review revealed five main subcategories:

1. *Task precedence models* such as PERT/GERT (Taylor and Moore 1980) and the Applied Signposting Model (Wynn et al. 2006) represent interactions between tasks in terms of information flows that define sequences. A relationship between two tasks indicates that the downstream task cannot be attempted until the upstream task has been completed, or progressed by a specified amount.
2. *Task dependency models* such as the design structure matrix (Eppinger et al. 1994) indicate where one task depends on information produced by another. Tasks in design and development often form interdependent clusters, such that there is no obvious sequence to complete them. A dependency model describes such interdependencies but does not indicate how they can

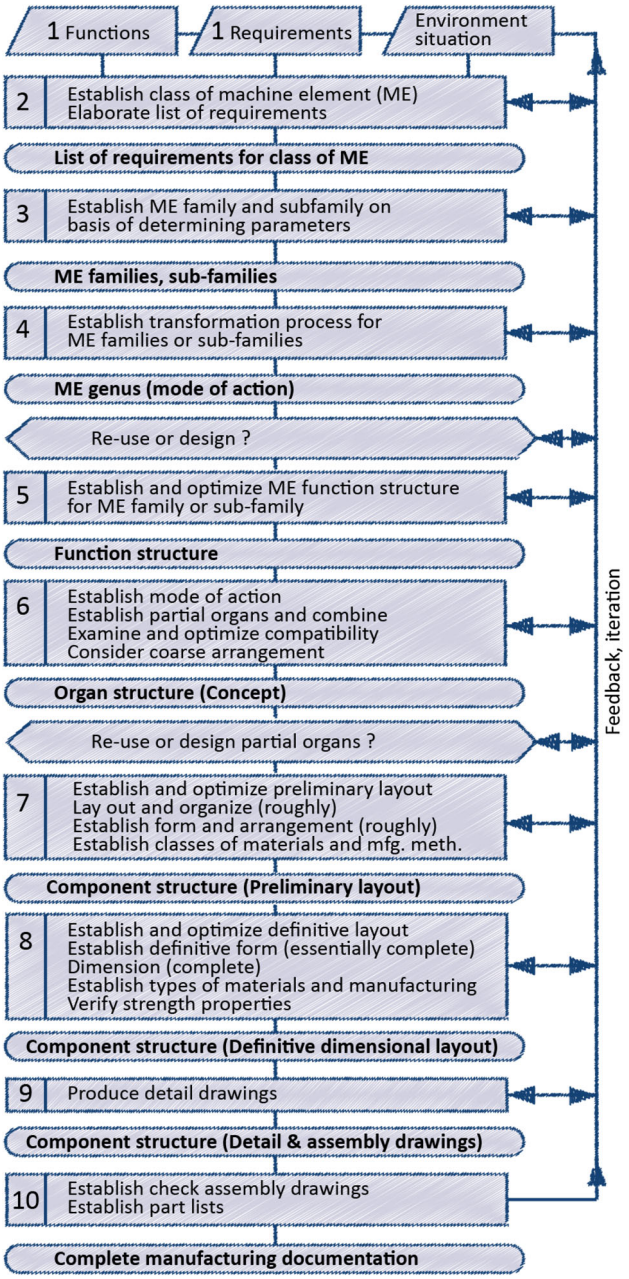

Fig. 7 General procedural model for designing of novel machine elements. Figure and caption reproduced from Hubka and Eder (1996) with permission of Springer

be resolved. Possibilities could include making initial estimates for some information and then iterating the tasks until convergence; or undertaking the tasks concurrently with frequent information exchange.

3. *Rule-based models* such as the adaptive test process (Lévárdy and Browning 2009) represent the DDP as a situated process in which tasks depend on rules concerning their context.
4. *Domain-integrating task network models* such as the multiple-domain matrix (Lindemann et al. 2009) explicitly focus on interactions between a meso-level flow of tasks and other information domains, such as design information.
5. *Agent-based task network models* such as the virtual design team (Levitt et al. 1999) consider how the meso-level flow of tasks is embodied in interactions between the people who participate in the DDP.

The following subsections discuss these subcategories in turn. For further information on models in this category, the reader is referred to Browning and Ramasesh (2007).

4.2.1 Task precedence models

Perhaps the most commonly used analytical modelling approaches in practice are those that represent processes as flowchart diagrams. Such approaches can be especially helpful for understanding, communicating, and reengineering processes (Melão and Pidd 2000). For instance, the event-driven process chain (EPC) notation maps a process in terms of activities, events, and logic gates. An application to product development is reported by Kreimeyer and Lindemann (2011). Other similar notations include the business process modelling notation (BPMN), and the IDEF3 process description capture method (Mayer et al. 1995). These modelling approaches may be viewed as interchangeable in many circumstances. Although the graphical notations may differ, the basic principles and focus on providing a visual notation for developing process maps are very similar. Typically, such notations provide an array of elements and graphical symbols allowing a modeller to represent additional contextual information. Some software tools provide features to support the construction of large models, for instance creating variants of a process, specifying rules to validate process models, and splitting process models across multiple diagrams.

The modelling approaches discussed above are generic and can be applied in many contexts. Focusing specifically on engineering design, Park and Cutkosky (1999) develop the design roadmap (DR) approach for modelling mature engineering design processes comprising many tasks with interactions that can be represented at multiple hierarchical levels. DR defines tasks in terms of explicit input and output information, because in comparison to representing this implicitly using arrows between tasks, "the description is more complete, the boundaries between tasks are defined, and a basis for developing interfaces between tasks is established" (Park and Cutkosky 1999). A noteworthy feature of DR is that it distinguishes between various types of relationships between tasks. First, strong precedence relationships strictly constrain the sequence of tasks.

Second, weak precedence relationships indicate flows that may or may not occur, e.g., relating to iteration. Third, side-effect relationships indicate where a task is not necessary to produce information, but may impact it when executed. Finally, constraint relationships indicate interactions between the information produced by tasks.

A limitation of the above methods is that they provide essentially static pictures, while processes typically involve "complex interactions that can only be understood by unfolding behaviour through time" (Melão and Pidd 2000). Researchers have accordingly developed computable models to study these issues using task precedence networks. The early work focused on application of program evaluation and review technique (PERT) to lay out the plan of work for development projects and then to focus management attention on the critical path (Pocock 1962). A related approach called critical chain/buffer management (CC/BM) is concerned with analysing a PERT-type network to identify buffers that protect the critical path and are used up as delays accumulate, so that those buffers can be actively managed (Herroelen and Leus 2001). PERT and CC/BM are based on acyclic precedence networks and do not explicitly account for iteration, which is one of the key characteristics of the DDP. Researchers considering this limitation applied later developments of PERT, namely graphical evaluation and review technique (GERT) (Pritsker 1966; Nelson et al. 2016) and its variant Q-GERT (Taylor and Moore 1980) to analyse DDPs under the assumption that some tasks in the network may trigger iteration probabilistically.

Other DDP modelling approaches explicitly represent dynamic flows of information in a process using variants of the Petri net. This is a formal approach which, in its simplest form, represents a process in terms of a network of places and transitions (Van der Aalst 1998). A transition is triggered when tokens accumulate in its input places, whereupon those tokens are absorbed and reappear in the transition's output places, potentially triggering other transitions in turn. Appropriately constructed Petri nets allow the dynamic behaviours of serial, parallel, and iterative task patterns to be modelled. For instance, McMahon and Xianyi (1996) use a Petri net-based process model as the basis of an automatic controller which directs computer processes to design a crankshaft. A shortcoming of the Petri net is that logical problems such as deadlocks can appear if the net is not appropriately structured, which becomes more difficult to achieve as the complexity of information flows and the number of possible routes increases. Considering these problems, Ha and Suh (2008) develop a set of Petri net templates that each represent a certain pattern of DDP task interactions. Larger models can then be assembled from these templates. Another issue is that, in the DDP context, it is common that changes in the planned process are required during its execution. This is also difficult to handle using Petri nets. Karniel and Reich (2011) address this issue with an approach to automatically generate or update a Petri net from a Task DSM (discussed in Sect. 4.2.2) in a way that ensures its logical correctness, thereby allowing simulation of a dynamic process involving complex information flows.

A more descriptively elaborate but less formal computable model based on a graphical precedence approach is the applied signposting model (ASM) developed by Wynn et al. (2006). The ASM is based on a hierarchical flowchart representation intended to be scaleable and familiar to practitioners. Similar to DR, tasks are specified in terms of input and output information, different dependency types can be represented, and an abstraction hierarchy of tasks and design parameters is provided with tool support to automatically generate simplified views (Wynn 2007). The ASM simulation algorithm was developed to handle processes having multiple intertwined iteration loops, which are difficult to configure in many other approaches. It also allows flexible specification of individual tasks' behaviours. In contrast to notations such as EPC and BPMN, flow logic such as AND/OR/XOR gates is not represented graphically, because this was found to require large and complex diagrams even for relatively simple processes. Instead, such logic is embedded in the tasks' configurations. The ASM was developed and applied through industry collaborations in the aerospace sector. For example, Kerley et al. (2011) describe how the approach was applied to model and simulate jet engine conceptual design in Rolls-Royce to support integration of improved lifecycle engineering tools into the process. Hisarciklilar et al. (2013) and Zhang et al. (2015) apply the approach to determine how to reduce process span time at Bombardier Aerospace. The ASM also laid groundwork for approaches to predict change propagation in a design process (Wynn et al. 2014), to analyse changes to the process itself (Shapiro et al. 2016), and to optimise resource allocation (Xin Chen et al. 2016).

A strength of graphical task precedence approaches is their intuitive flowchart-style notation which can be easily understood by most people. Another is the flexibility; a model may be constructed at different levels of rigour and formality according to the modeller's needs and preference. However, such models also have limitations. As is apparent in the example of Fig. 8, although it is possible to model quite complex processes, graphical network models do become unwieldy as a model's structure becomes more complex and incorporates more concurrent tasks, because it becomes difficult to visually arrange and make sense of the many information flows required. Flows that connect across a long distance of the model are especially difficult to read and manipulate. The effort to make changes to a

graphical model tends to increase substantially with the model's scale and density, due to the need to manually reorganise the layout and rewire the connections. Some of these difficulties may be partially addressed by organising a model hierarchically into subprocesses, but this can introduce further challenges in managing and visualising connections that cross levels and can also cause problems if the hierarchy later needs to be repartitioned. Another consideration is that if a model is used for simulation, some schemes require careful configuration and painstaking verification to ensure it operates as intended in all scenarios, especially if it incorporates a dense structure of dependencies with concurrent flows and intertwined iteration loops (Karniel and Reich 2009).

ProModeller is a task precedence approach which is not based on node-arrow diagramming and thus avoids some of these issues. This system allows modellers to represent a process by hierarchically combining process elements drawn from a standard library comprising around 50 objects (Freisleben and Vajna 2002), each representing either a type of task or a structural element. Tasks can be configured when instantiated into a model. Structural elements are essentially hierarchical containers that specify the procedure for attempting the objects nested within: sequentially; in a cycle of iterative refinement; concurrently; or by selecting one from a set of alternatives (Vajna 2005). The reflection of process behaviour in structure ensures that models constructed using this approach are logically correct. In consequence, it is not necessary to validate a model's structure prior to analysis. This may facilitate the distribution of modelling effort among many process participants. On the other hand, in comparison to graphical network approaches, tree-structured approaches like ProModeller provide less flexibility for modelling complex information flows and arguably a less visually intuitive representation.

The task precedence models discussed in this subsection may be especially useful where design processes are relatively routine, while also involving enough complexity that stakeholders may not fully understand them prior to modelling. These situations do often occur in practice—for instance in the evolutionary development of large-scale designs (Wynn et al. 2014). The situated and responsive aspects of designing may be embedded in the possibility of some tasks triggering iteration, or may occur within individual tasks and thus be below the level of resolution of a model. They may also render a model inaccurate if they lead to changes in the tasks that are needed or in the way information flows between them.

4.2.2 *Task dependency models*

To recap, task dependency models represent the information dependencies between tasks as well as, or instead of, a procedure for attempting them. Such models emphasise that the tasks could be organised in several ways. For example, they could be attempted in different sequences or in parallel. Approaches which incorporate dependency

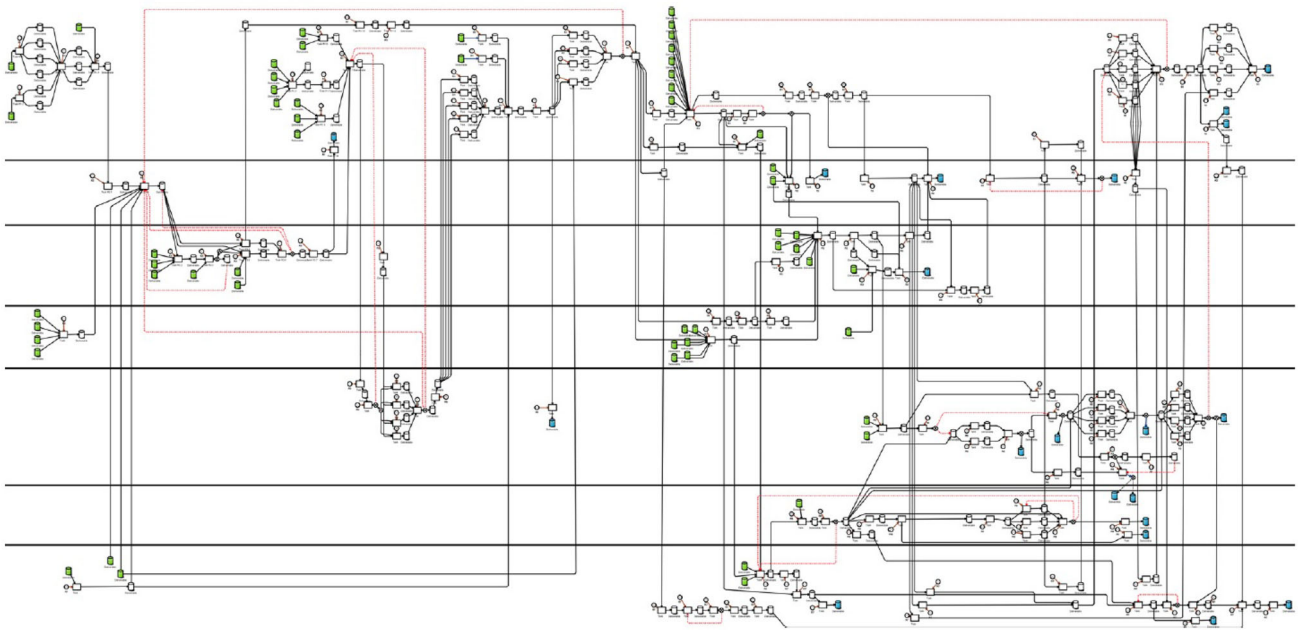

Fig. 8 Task precedence model of a partial jet engine conceptual design process. Swimlanes are used to represent design responsibilities. Reproduced from Wynn et al. (2014) with permission of ASME

models are based on the premise that a process can be improved by studying the underlying structure of the situation.

The most well-known model in this category is probably the design structure matrix (DSM) introduced by Steward (1981). A DSM is a square matrix in which a mark in a cell indicates that the element in the row depends upon that in the column (see the example in Fig. 9). Where the elements represent tasks and the connections represent information dependencies, the matrix is called a Task DSM (Eppinger et al. 1994). If all the marks lie below the leading diagonal in one or more of the possible orderings of the rows and columns, the process may be completed by attempting tasks sequentially or in parallel. Conversely, if it is not possible to find such an ordering, some of the tasks are interdependent and iteration may be required to resolve them (Eppinger et al. 1994). Algorithms have been developed to analyse a DSM to examine or exploit such structural characteristics. The algorithms include: sequencing, which is attempting to find a lower diagonal reordering, i.e., a sequence of tasks to minimise information feedback and, therefore, reduce the possibility of iteration; banding, identifying independent elements in a sequenced DSM, i.e., tasks which may be attempted in parallel; and clustering, attempting to group elements into strongly connected sets with low inter-cluster connectivity, i.e., groups of tasks that may be appropriate to perform essentially in isolation (e.g., Kusiak and Wang 1993b; Yassine 2004).

The Task DSM has been extensively adopted in research literature as the basis of models to analyse DDP characteristics, especially those related to decomposition and integration. The key consideration here is that when a high-level task such as designing a system is decomposed into subtasks that will be undertaken by different people or teams, interdependencies are invariably created between those subtasks. It is, therefore, important to carefully organise the subtasks and manage the information flows between them to minimise the rework that might be generated when tasks' outputs are reintegrated—especially if some of the work will be done concurrently. One seminal meso-level model considering these issues is the work transformation matrix (WTM) developed by Smith and Eppinger (1997a). The WTM focuses on situations in which interdependent tasks are executed in parallel with frequent information transfer to manage their interdependencies. It assumes that each task in such a group continuously creates iteration work for the others that depend on it, at a constant rate. The dependencies and their corresponding rates are represented in a Task DSM. Smith and Eppinger (1997a) show how eigenstructure analysis can be used to identify the drivers of iteration within a coupled task group if the WTM assumptions hold. Assuming instead that tasks are executed in sequence, such that each task might create rework for others already completed if a dependency exists between them, Browning and Eppinger (2002) build on the earlier work of Smith and Eppinger (1997b) to develop a Monte Carlo simulation model which they use to evaluate the cost and schedule risk associated with different task sequences and thereby identify the best sequence for a given task decomposition. These two models, respectively, described as parallel and sequential rework models, have influenced many other research articles (e.g., Bhuiyan et al. 2004; Cho and Eppinger 2005).

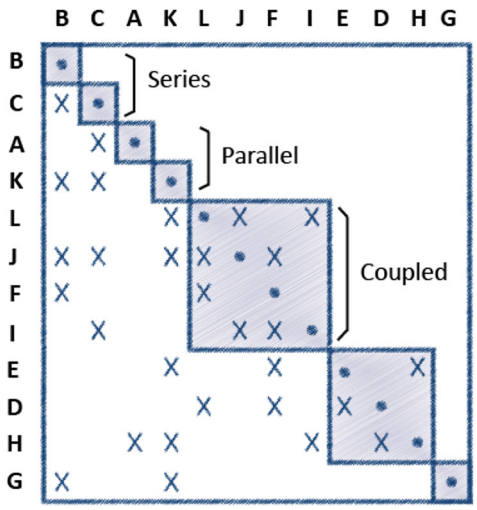

Fig. 9 Binary design structure matrix, partitioned to represent a sequence. Figure and caption reproduced from Eppinger et al. (1994) with permission of Springer

The Task DSM provides a compact notation which can be especially useful for processes involving dense structures of information dependency. It is also useful to concisely visualise the properties of different dependencies, if meaningful symbols and/or numbers are placed in each cell (Browning 2016). Achieving a comprehensible visual layout is likely to be easier than when graphical networks are used. Another advantage is that the approach can be applied without specialised software. Many computations can be expressed and programmed as operations over the matrix cells. On the other hand, some weaknesses are also apparent. DSMs are not well suited to convey detail, and thus, it can be easy to misplace marks when constructing or reading large matrices. It is not clear how to deal visually with opening and closing hierarchical structures in a DSM model. Sequential and parallel flow structures are difficult to visualise (Park and Cutkosky 1999), because, although clusters of tasks can be easily indicated as shown in Fig. 9, there is no equivalent of swimlanes. More information on the Task DSM and the many related models can be found in Eppinger and Browning (2012) and the review article by Browning (2016).

Another established dependency modelling approach is IDEF0, which uses a hierarchically structured set of diagrams to represent a system in terms of functions and the interactions between them (USAF 1981). Applied to the DDP, functions are in essence similar to tasks. Each IDEF0 diagram comprises between three and six functions, which are represented as boxes and interconnected by labelled arrows. Arrows indicating a function's inputs enter at the left of the box, and are transformed to produce outputs which leave from the right of the box. Control arrows enter the top of a box and indicate constraints on the function's operation. Mechanism arrows enter the bottom of a box and indicate provision of a means for executing the function. Any function box can be decomposed into a more detailed diagram showing its subfunctions. Functions can be linked across and between levels in the hierarchy, and the model may include a glossary of terms (USAF 1981). In comparison to DSM, the IDEF0 approach is more expressive, but less concise. A large set of diagrams is often needed, which can be time-consuming to produce (Colquhoun et al. 1993).

Although not as prominent as DSMs in the research literature, IDEF0 has been quite widely applied for DDP modelling. For example, Kusiak et al. (1994) discuss its use to support reengineering of design and manufacturing processes, arguing that the notation can help with perceiving a process at different levels of detail and with exploring how the constraints on a task's execution can be relaxed. ADePT PlanWeaver is a planning support tool for the construction industry which is based on an IDEF0-style representation, enhanced to indicate the discipline associated with each flow into a task, as well as the strength of the dependency (Austin et al. 1999). In the approach, a library of generic construction processes is used to construct a customised process model for a specific project, which can be viewed as a Task DSM and then sequenced to minimise the scope of cycles that may cause iteration. Identifying dependency loops that remain and finding ways to eliminate them, for instance by splitting some tasks into several parts, allows the project to be sequenced and a schedule to be produced (Austin et al. 2000). More recently, Romero et al. (2008) introduce an enhanced IDEF0+ approach. This includes additional symbols to distinguish the main flow of information from other interactions, such as coordination and cooperation, that are needed in a collaborative design process.

To summarise, the main advantage of task dependency models is their emphasis on information flow constraints rather than procedures—because understanding constraints is helpful when constructing a plan or seeking opportunities for process improvement. On the other hand, Austin et al. (1999) identify one disadvantage in that untrained readers tend to incorrectly assume a task sequence.

4.2.3 Rule-based models

Task precedence and dependency models as discussed above view DDPs as essentially similar in nature to other business processes, albeit with a high level of uncertainty and with the expectation of iteration. One criticism that might be levelled at such models is that they attempt to represent design processes but do not explicitly integrate an important insight gained from research into the nature of design activity—its situatedness (see Sect. 3.3). Rule-based models offer a possible route to address this limitation. They aim to model how process outcomes emerge through the interaction between the rules that define task properties and the design situation which changes as tasks are executed.

Some meso-level work in this area built on the Signposting approach of Clarkson and Hamilton (2000), which was discussed in Sect. 3.2. This model was extended through a series of Ph.D. projects to study the multitude of routes that might be possible in a complex, concurrent design process. Features added to the model to do this included: a probability density function defining the duration of each task; multiple outcomes from each task with a probability of each occurring; and resources required by each task along with their limited availability (O'Donovan et al. 2004). Among other insights this model, called Extended Signposting, was used to show how both the probability and desirability of each route should be considered when planning a design process. The adaptive test process (ATP) takes a similar approach, viewing a DDP as a complex adaptive system that emerges from a "primordial soup" of activities together with rules governing their selection (Lévárdy and Browning 2009). In comparison to Signposting, ATP offers more concrete criteria for selecting tasks, considering their roles in driving technical performance measures (TPMs) closer to specified targets. Lévárdy and Browning (2009) argue that at each step, the next task should be selected to maximise expected project value in terms of the TPMs, time, and cost. The ATP incorporates a simulation model that can be used to examine the value generated by different tasks and activity modes at different points in a project, among other contributions (Lévárdy and Browning 2009). More recently, Wynn et al. (2011) describe a process model in which key properties of tasks are defined according to rules that consider evolving uncertainty levels relating to design information. To illustrate, the time spent on an FEA task would be influenced by the expected accuracy of boundary conditions, which would propagate through the task to influence the expected accuracy of its outputs. In this model, a design is progressed through iterative cycles which continue until uncertainty levels converge to acceptable values. Wynn et al. (2011) suggest that this

approach can be used to explore how different facets of design uncertainty may contribute to project delays.

Apart from the possibility of capturing a process' interdependency with the evolving situation, a noteworthy feature of Signposting and ATP in particular is that they in principle allow models to be constructed from knowledge of individual tasks or process fragments, because an information flow network does not need to be explicitly represented. This bypasses the requirement for an integrated overview of the process, which can be difficult to develop in practice. On the other hand, when compared to the approaches discussed in the previous two subsections, rule-based models are difficult to visualise and it is not clear how to validate all possible routes they allow. Research towards addressing these limitations is reported by Clarkson et al. (2000). For the moment though, such models remain mainly of academic interest.

4.2.4 Domain-integrating task network models

Domain-integrating task network models explicitly integrate process models capturing an end-to-end flow of tasks with detailed information about other domains such as the product being designed. Eckert et al. (2017) argue that such models could be useful to guide trade-offs between design characteristics and process performance. For example, they might help to decide whether design changes should be accepted during a project, considering whether the design improvements would justify the additional time and effort in the development process.

Recently a lot of attention has been paid to domain-mapping matrices (DMMs) and multiple-domain matrices (MDMs). These are extensions to the DSM which allow modelling of linkages between different types of element (Kusiak and Wang 1993a; Danilovic and Browning 2007; Lindemann et al. 2009; Bartolomei et al. 2012). Danilovic and Browning (2007) discuss application of DMMs to explore connectivity between the process domains of tasks, components, and teams. By analysing the domains independently and in combination, it is possible to identify mismatching structures. For example, a team structure which does not reflect the decomposition of tasks in the process may contribute to communication overhead or rework (Kreimeyer and Lindemann 2011). Sosa et al. (2004) discusses how such structures can be identified using a DMM approach, and how this can be used to focus coordination effort on the interactions which are likely to drive design change and iterations. A key aspect of MDM methodology is the use of filter operations to derive indirect dependencies, for example, computing an implied Task DSM from an MDM showing the tasks' inputs and outputs (Lindemann et al. 2009). Another element is using graph-theoretic metrics such as betweenness centrality and cycle count to develop insights about the importance of nodes and patterns in the network (Kreimeyer and Lindemann 2011). Lindemann et al. (2009) and Kreimeyer and Lindemann (2011) define a standard set of domains (e.g., subsystems, tasks, resources, etc) which can be modelled in a development project and a set of metrics and filters for analysing models thus constructed.

Object-process methodology (OPM) provides an integrated representation of processes and objects using a formal graphical notation or equivalent formally structured sentences (Dori 2002). A model constructed using OPM comprises a hierarchically organised set of object-process diagrams (OPDs) that represent both processes and their related objects (ISO/PAS19450 2015). Several types of structural link allow the modeller to connect diagram elements within the process domain or within the object domain, while several types of procedural link can be used to connect elements across these two domains. OPM is a general-purpose methodology that has been applied in different contexts. Of particular interest to this review, Sharon et al. (2013) consider how it can support planning and control of development projects, by clarifying how the project tasks (modelled as processes) are interrelated with the required resources and the hierarchy of deliverables (modelled as objects). In their approach, a project is decomposed into a hierarchy of tasks and deliverables, considered concurrently. The OPM representation is then analysed to generate summary views useful for project management. Sharon and Dori (2015) further develop this method, arguing that it could help to avoid mismatches and inconsistencies between the models and documents used to manage a project.

Other models integrating product and design process information have been developed with the specific objective to support resolution of conflicts among design parameters. For example, the DEPNET approach stipulates modelling a process as it unfolds, along with the design information associated with each task (Ouertani and Gzara 2008). The resulting trace constitutes a network of dependencies among information items, which can be used to assess the knock-on impact of design changes. A similar approach is taken in CoMoDe, an object-oriented model intended to maintain a trace of the model versions that are created and used at each step in a collaborative design process (Gonnet et al. 2007). CoMoDe represents a hierarchy of process activities and constituent operations; requirements; the actors who perform each activity; characteristics of the artefact as it is evolved; and decision rationale. Gonnet et al. (2007) describe how it can be applied to detect conflicts among models that exist simultaneously in the collaborative design process, according to the logic by which those models were generated. Overall, process-oriented conflict management seems a theoretical

approach involving step-by-step capture of design history using rather complex representations. Although the potential is demonstrated by examples, the respective authors do not report evaluation of the proposed support tools in an industry context.

Focusing on coordination in major projects, Rouibah and Caskey (2003) develop an engineering work flow (EWF) approach based on identifying the engineering parameters whose values need to be determined—this can be partly constructed by reference to similar past projects. The parameters are linked into a network to represent their interdependencies, which can evolve during a project. Parameters are also linked to the responsible parties. During design, parameter values are iteratively developed through increasing "hardness grades". Six steps are defined to transition between successive hardness grades, to ensure that the change is coordinated among impacted parties. This approach seems to have strong potential to support the coordination tasks to ensure consistency and transparency during a project. However, it does not describe the specific engineering tasks required to determine each parameter's value.

In comparison to the approaches reviewed in Sects. 4.2.1, 4.2.2 and 4.2.3, domain-integrating models more strongly emphasise how a DDP interacts with its context. While this potentially offers more insight, it also requires more information. Consequently, it may be difficult to create large-scale models in such approaches and ensure their consistency (Park and Cutkosky 1999), as well as to visualise and understand the models once created. There are many other approaches in this category. For focused reviews of integrated models and further discussion of their advantages and limitations, the reader is referred to Eckert et al. (2017) and Heisig et al. (2014).

4.2.5 Agent-based task network models

Finally, agent-based models (ABMs) have been developed that combine meso-level task relationships with micro-level models of agent behaviour. Such models offer the possibility to study factors impacting a process in a more realistic context than the other models described in this section. For instance, they can incorporate factors such as organisational structures and the many non-design activities that project participants must attend to—such as going to meetings, chasing colleagues for information, and other coordination activity that emerges as a project unfolds.

In one influential example, the virtual design team (VDT) developed by Cohen (1992), Christiansen (1993) and colleagues represents individual designers and managers in a project as information-processing agents. These agents interact by generating and responding to messages according to rules. Messages can involve passing design information between tasks and also the handling of exceptions, which occur when an agent must stop work and seek more information before they can complete their assigned task. In the model, message handling depends on factors such as the organisation structure and communication tools available. Later developments of the VDT accounted for additional influences such as incongruency between actors' goals. Levitt et al. (1999) discuss a case study of satellite launch vehicle design, in which the VDT was used to evaluate the impact of proposed changes such as increasing individuals' skill levels and improving alignment of their objectives. Other ABMs developed for the DDP context include the Agent Model for Planning and rEsearch of eaRly dEsign (AMPERE), which focuses on studying the impact of requirements changes during design (Fernandes 2015), and the model of Crowder et al. (2012), which focuses on the factors involved in effective team working.

Some advantages of ABMs were discussed at the start of this subsection. In addition, it may be noted that ABMs can represent the decisions of situated actors and thus may be well suited to account for the responsive and emergent facets of the DDP (Garcia 2005). In terms of disadvantages, developing an ABM requires complex configuration or programming of a specialised tool and may be beyond the reach of many would-be modellers. Second, the models are each unique and do not lend themselves to graphical representation. As a result, their mechanics can be opaque except to their creator, which might lead to credibility concerns. Finally, although ABMs might be helpful to build understanding of the factors influencing DDP performance, they cannot easily be used to document or prescribe a process.

4.3 Meso-level abstract models

Abstract models on the meso-level provide conceptual frameworks for understanding how meso-level process flows, or models of them, relate to the design's progression. In contrast to other categories of meso-level model, they do not specify or analyse tasks in detail.

Some abstract models conceptualise the design process as a series of tasks that transition in a progressive way between the different types of information or knowledge that are used as a design is created. Many of these are informed by the early work of Hubka, Andreasen, and others who showed how a mechanical design can be described as a structure of information that cuts across different "domains". Overviews of the product-focused aspects of this work can be found in Buur (1990), Andreasen (2011), Eder (2011), and Hubka (1982). Applying these concepts to the design process, Theory of Domains (recently summarised by Andreasen 2011;

Andreasen et al. 2015) contends that designers consider an emerging mechanical design from four perspectives or domains: (1) a process of transformations effected by the product in use; (2) functions that provide those transformations; (3) organs which provide physical effects required for functions, through interaction between parts; and (4) physical parts themselves. The theory states that designers establish these domains in the sequence listed above, noting that stepping back and forth between them is also likely (Buur 1990). Within each domain, a design is described by multiple product models that can each be categorised on a two-dimensional grid: abstract vs. concrete, and simple (undetailed) vs. total (detailed) (Fig. 10). Micro-level procedural models such as those reviewed in Sect. 3.1 can be seen as either assisting work within a domain or guiding transitions between domains (Buur 1990).

Related to the theory of domains, Grabowski et al. (1996, 1999) develop the Universal Design Theory (UDT) based on the concept of design working spaces (DWSs). Each DWS is bounded by constraints that determine how it fits into a higher level system, and comprises the design's elements and relationships that are developed through four stages, namely requirements, functions, physical principles, and parts. The design process is seen as a series of operations in which a solution is progressively developed within its DWS by stepwise moves that can be categorised on three dimensions. The first dimension is concretisation vs. abstraction. For example, concretisation might move a solution state from functions to structures, while abstraction might move it in the opposite direction. On the second dimension, detailing vs. combination, a problem is decomposed into subproblems with their own DWSs, or subsolutions are combined into higher level solutions. On the third dimension, variation refers to searching for alternative solutions on the same level of abstraction, while its counterpart, limitation, refers to adding constraints that reduce the solution space. Grabowski et al. (1996) also emphasise the importance of guiding the process from one step to the next.

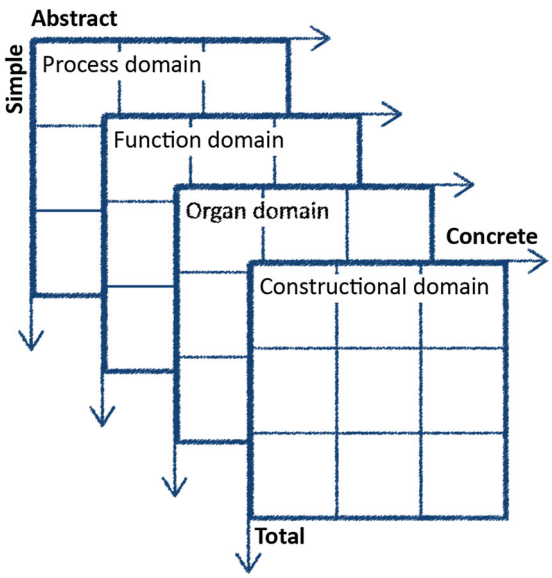

Fig. 10 Theory of domains views design as a process in which design information is established through increasingly concrete domains. It provides a framework in which models and methods used during design can be positioned. Reproduced from Andreasen (1980) with permission of the author

Finally, characteristics-properties modelling/property-driven development (CPM/PDD) was developed to provide a theoretical framework for integrating computer tools into the design process and vice versa (Weber et al. 2003). CPM/PDD states that a design comprises characteristics, which are set by designers, and properties, which describe the design's resulting behaviours. A design process is presented as a collection of synthesis tasks, which determine or create characteristics from desired properties, and analysis tasks, which determine properties from characteristics. The model suggests that tasks are also influenced by external conditions, such as load cases, and can be supported through prescriptive methods such as those discussed in the previous sections. Key features of design that the model aims to encompass include: how the process is driven by the difference between desired and real properties; how the product definition becomes more complete over time as more characteristics are created and their values determined; how partial solutions can be integrated into an emerging design; and how iterations may be caused by conflicts, e.g., when multiple synthesis tasks affect the same properties (Weber 2014).

4.4 Meso-level MS/OR models

Meso-level models of the fourth and final type, MS/OR, are similar in many respects to the meso-level analytical models discussed in Sect. 4.2. The key distinction is that models in this category are created as mathematical or computational tools for research in which representative or synthetic cases are analysed to extract general insights—whereas the analytical models discussed earlier provide approaches that practitioners might in principle use to model, analyse, and improve their specific situations.

One stream of work in this category focuses on developing mathematical models to study how concurrency may help to reduce lead time by bringing more resource to bear, at the cost of increased rework. For example, AitSahlia et al. (1995) develop algebraic models that show how the number of tasks that have to be redone if iteration occurs increases as their concurrency increases. Their models demonstrate how the tipping point at which further increases in concurrency start to yield increases instead of

reductions in process duration is determined by the probability of each task creating rework for others. Hoedemaker et al. (1999) consider a similar situation, developing models to explore how the increased need for communication and the need to reintegrate tasks cause additional efficiency losses as concurrency is increased. Other authors consider design reviews. For example, Ha and Porteus (1995) develop a mathematical model to study the optimal timing of such reviews during concurrent product and process design. In this model, the desirable effects of frequent design reviews are to find flaws before they are incorporated into the design, and to validate interim product design work so that it can be released to process design, enabling concurrency. This is set against the time required to set up and execute the reviews. Ha and Porteus (1995) show that the optimal frequency of reviews depends on whether the concurrency or quality issues dominate. Their model is extended by Ahmadi and Wang (1999) to also consider how resource is allocated to different design stages. In this case, the model is used to consider how the reviews should be scheduled with a view to minimising the risk of missing targets. A number of other MS/OR models focus on managerial decisions relating to stage overlapping, without explicitly representing the interactions among numerous discrete tasks—these are accordingly categorised as macro-level and discussed in Sect. 5.4.

Another group of models emphasise how the task decomposition influences convergence of a concurrent, iterative design process—as explained by Browning (1998), "tightly coupled, highly iterative processes can expect greater difficulty converging to an acceptable design under a given schedule and budget". Considering this issue, Yassine et al. (2003) develop an MS/OR model to study the causes of oscillatory situations in which progress is repeatedly thought to be on schedule before falling behind, arguing that this causes several knock-on problems such as short-termism in resource allocation. They use their model to show that this situation arises because teams that work concurrently on interdependent problems only coordinate periodically and thus often make design decisions based on outdated information. Braha and Bar-Yam (2007) focus on structural characteristics of the information flow network among tasks being worked concurrently. They develop a model considering that when any task is solved, it is possible that this will cause any interdependent tasks to require iteration. They analyse task networks from several domains and find there are common characteristics. In particular, most tasks are not strongly connected, but those that are strongly connected are shown to be especially susceptible to such iterations. Other researchers have studied convergence problems using spectral analysis, developing MS/OR models based on the Work Transformation Model (WTM) developed by Smith and Eppinger (1997a, see Sect. 4.2.2). In one such model, Loch et al. (2003) apply an eigenstructure analysis to show that convergence of an iterative process becomes less probable and more time-consuming as the number of coupled tasks increases. In two others, Huberman and Wilkinson (2005) and Schlick et al. (2013) create spectral models incorporating fluctuations in task performance, both showing that variance in overall process time can increase dramatically if the fluctuations exceed a certain threshold.

The above models consider a process in terms of tasks only, without reference to characteristics of the emerging design. In contrast, Mihm et al. (2003) describe a model of design convergence in which decision-making considering design trade-offs is explicitly represented. Their model represents a design situation as a network of interconnected components, each defined by a single design parameter. Every design parameter should be chosen to minimise a performance parameter for the corresponding component. However, a component's performance depends not only on its own design, but also on the designs of all components connected to it. The model simulates how iteration can be used to converge on a solution, through a series of steps in which all parameters are updated simultaneously. Running simulations based on randomly generated data sets, Mihm et al. (2003) show that convergence takes longer with larger problem sizes and eventually becomes impossible. They develop recommendations to improve the speed of iterative convergence: ensuring designers aim for the global performance function instead of optimising locally; accepting a slightly lower level of performance overall; minimising information transfer delays so that decisions are based on up-to-date information; converging step-by-step towards the desired outcome, e.g., by exchanging preliminary information through a series of iteration cycles; and structuring the design into relatively independent modules.

Overall, the models discussed in this subsection, and others in the same category, are rather general in nature and do not offer guidance tailored to specific situations. However, researchers' conclusions from the models can provide useful insight into the drivers of (desirable or undesirable) development project behaviours.

5 Macro-level models

This section completes the review by discussing the third and outermost of the three levels of the organising framework shown in Fig. 1. Many of these macro-level models concentrate on the large-scale organisation and management of design and development. Some consider interactions between the design and development process and the

context into which the design will be delivered. Considering the first of these two situations, the primary difficulty companies face is arguably the integration of systems, disciplines, tools, processes, and personnel (Andreasen and Hein 2000). Research addressing this is often known as Concurrent Engineering (CE) (e.g., Prasad 1996a) or Integrated Product Development (IPD) (e.g., Andreasen and Hein 2000; Vajna and Burchardt 1998). According to Prasad (1996a), CE emphasises approaches "to elicit the product developers, from the outset, to consider the 'total job' (including company's support functions)". Some of the key facets of this philosophy are to use advanced collaboration tools including approaches such as Quality Function Deployment (Hauser and Clausing 1988), appropriate team structures, and Design for X methods to increase concurrency and information exchange between coupled tasks, teams, and design considerations (Prasad 1996a; Vajna and Burchardt 1998). Overall, CE/IPD is thought to compress lead time and support integration by reducing the mistakes and oversights that can cause late design changes (Prasad 1996a).

5.1 Macro-level procedural models

A number of prescriptive models provide graphical depictions and explanations of the contextual issues that need to be addressed during design, ranging from production processes through to economic considerations. Examples include the IPD model (Andreasen and Hein 2000), the total design model (Pugh 1991), the concurrent engineering wheels (Prasad 1996b), (Fig. 11), and the model of engineering design set in context developed by Hales and Gooch (2004). Models of this type are discussed further by Wynn and Clarkson (2005).

Other models in this category prescribe DDP management structures and philosophies thought to mitigate the risk of costly loop-backs, i.e., iterations between stages of the development process. One such model commonly found in companies is the stage-gate process (Cooper 1990), which emphasises the use of formal, structured reviews to ensure a design is sufficiently mature before allowing it to proceed from one stage to the next (Fig. 12). Another is the Systems Engineering Vee model (Fig. 13) which graphically emphasises decomposition of a complex design into subsystems which are developed individually, and then integrated, verified, and validated at every level of the subsystem hierarchy (Forsberg et al. 2005; VDI2206 2004). Key concerns here include ensuring the proper definition, flowdown and control of requirements and interface definitions to avoid synchronisation problems and rework. A third model that has gained attention is set-based concurrent engineering (SBCE), which advocates controlled reduction of technical uncertainties through a focus on up-front learning about whether the design is feasible. The guiding principle is that choosing the right concept means fewer surprises later, reducing rework, and allowing more standardised, more efficient work later in the design process (Kennedy et al. 2014). SBCE proposes that this should be approached by developing and maintaining several workable designs for each subsystem, and gradually eliminating alternatives that are found to be infeasible or found to generate integration difficulties as the design moves forward (Fig. 14). This may be compared against the more common practice of creating one design for each subsystem and iterating until they can all work together. Authors have also considered how Lean models developed in manufacturing, involving concepts such as JIT and takt periods, can be applied to manage routine aspects of development processes (e.g., Oppenheim 2004). Holistic procedural models that incorporate Lean and SBCE include descriptions of the original Toyota Product Development System (e.g., Sobek et al. 1999; Liker and Morgan 2006); the learning first product development model of Kennedy (2008); and the LeanPPD model of Al-Ashaab et al. (2013).

The approaches discussed above focus on avoiding rework by establishing an essentially funneled structure in which the design space is progressively narrowed; decisions thought to have greatest consequence are taken earlier in the process and efforts are made to inform them as fully as possible. This overall strategy is visualised in the textbook model of Ulrich and Eppinger (2015). In contrast, agile models prescribe structured iterative cycles in which the design is repeatedly reintegrated as it progresses through increasing levels of definition (Cusumano and Selby 1997). This and other forms of iterative incremental development (IID) have been accepted in the software development context for some time (see Larman and Basili 2003 for a review) and have been proposed as possible approaches to managing product development as well (e.g., Turner 2007). They may be especially useful in contexts where customer needs or technology evolve rapidly, in cases where requirements are difficult to specify and where the emerging solution influences the nature of the problem. Considering similar issues, Ottosson (2004) developed Dynamic Product Development (DPD), a model targeted at projects involving substantial innovation and creativity. Ottosson (2004) argues that the traditional emphasis on controlling projects by formal documentation and review leads to delayed information and reactive management. He also highlights the difficulty of long-term planning in a project involving uncertainty. To address these issues, DPD prescribes delegation of control allowing continuous managerial involvement at all levels, which is thought to facilitate real-time dynamic guidance. Furthermore, DPD aims to minimise loop-backs

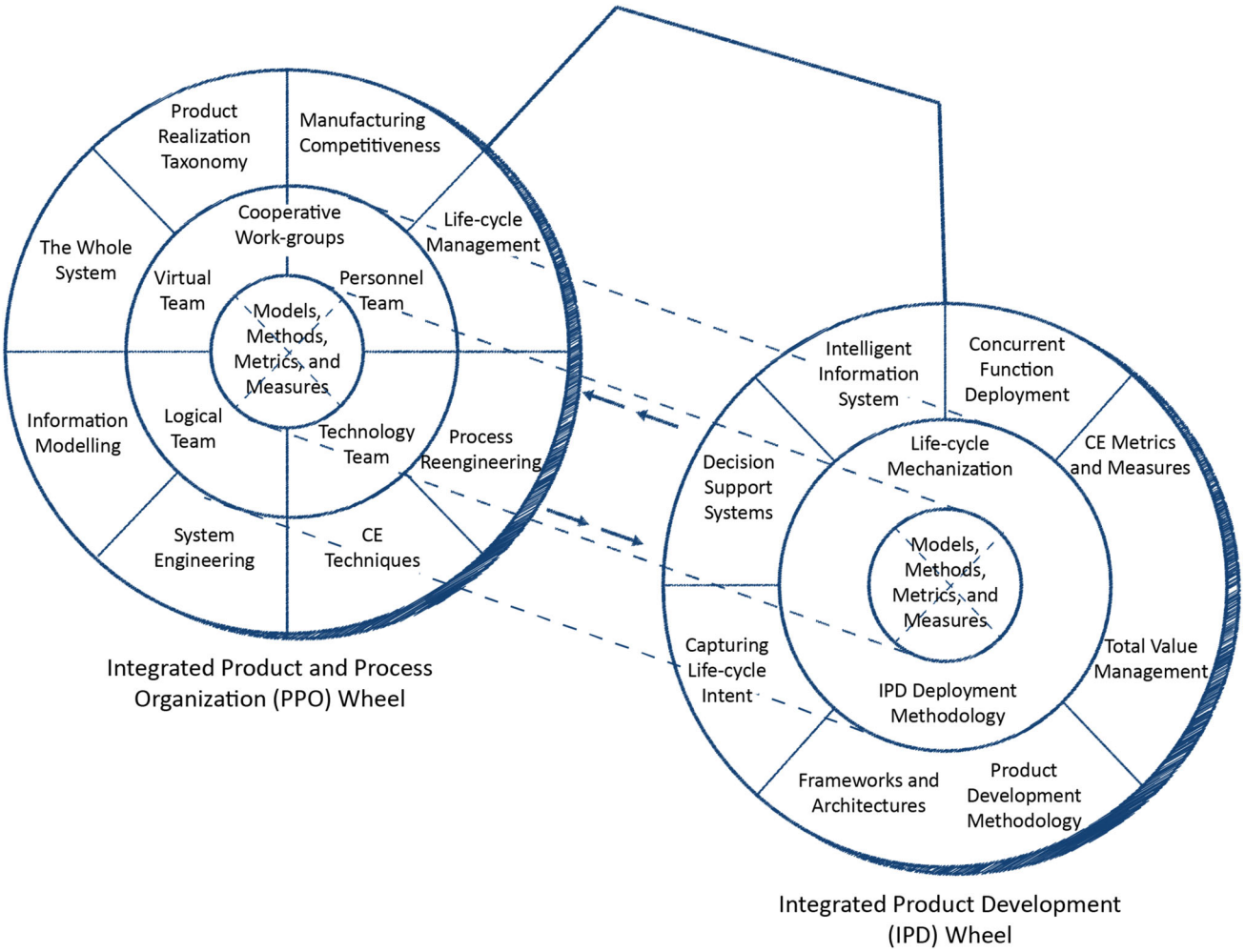

Fig. 11 Concurrent engineering wheels emphasise integrating the 'total job' in a CE project. The coupled wheel structure indicates that issues are simultaneously addressed and capabilities are developed in unison. Reproduced from Prasad (1996b)

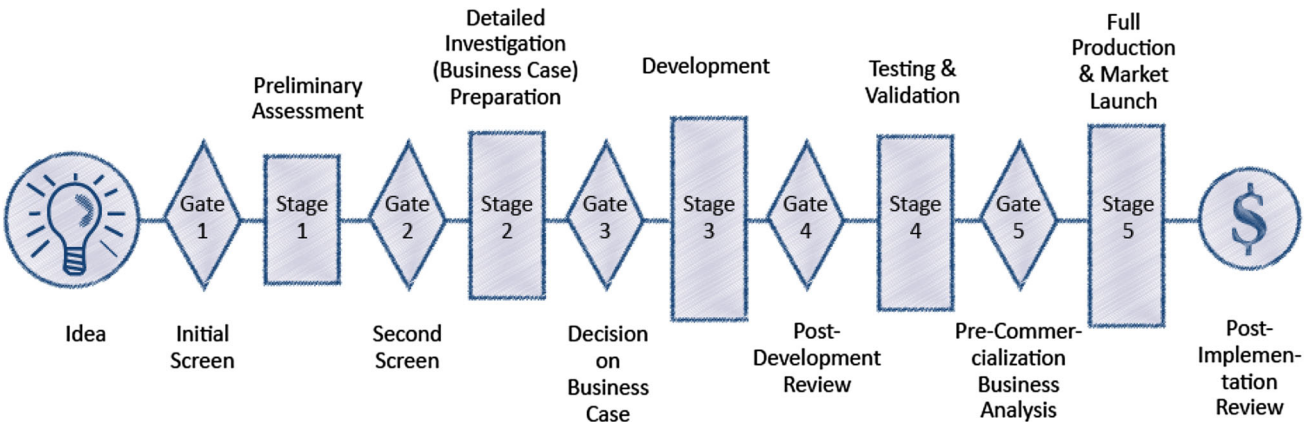

Fig. 12 Stage-Gate model emphasises the need to ensure that a design is sufficiently mature to exit each stage of the development process, to prevent costly loop-backs. Figure reproduced from Cooper (1990) with permission of Elsevier

by allowing the concept to be adjusted continuously throughout a project, rather than freezing it early. A key consideration regarding application of dynamic, iteration-driven approaches such as IID and DPD to large projects is ensuring sufficient discipline and control of the development process (Turner 2007).

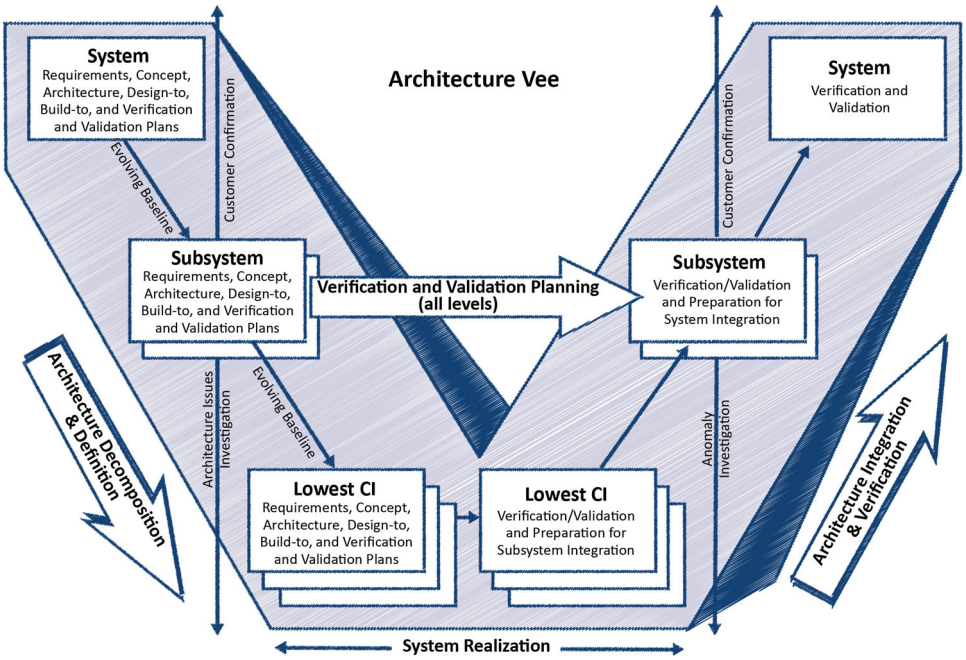

Fig. 13 Vee model of the systems engineering process emphasises management of decomposition and integration to avoid rework. Figure reproduced from Forsberg et al. (2005) with permission of Wiley. Copyright ©2005 by John Wiley & Sons, Inc

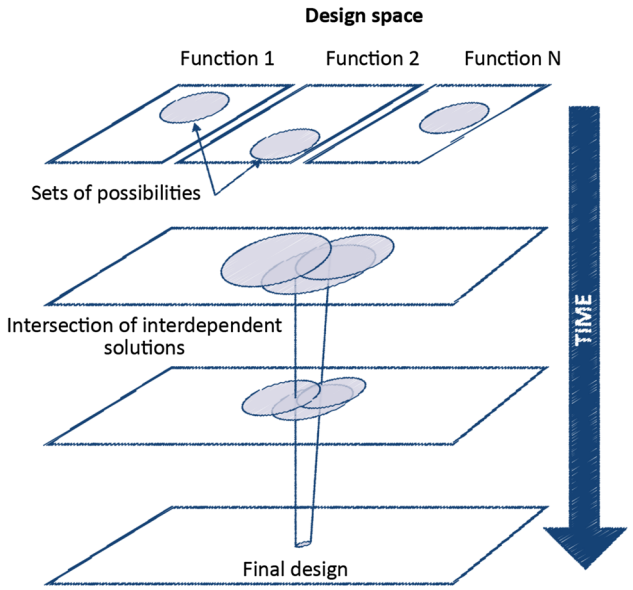

Fig. 14 Depiction of the set-based concurrent engineering process. Reproduced from Raudberget (2010) with permission

Practice often integrates characteristics of several models from this category without following any one exactly as prescribed. Maffin et al. (1995) argue that although such models can appear too general for easy application, they can be adapted to a particular context. They propose that a set of critical factors which define the organisation and the product are influential upon the product development process, and that classifying companies according to this framework could form the basis for guiding the selection of suitable models for a company.

One overall challenge with macro-level procedural models is handling their implementation in a particular company, each of which will start from a unique set of issues and existing processes. The high level of abstraction of the models arguably does not provide much guidance towards improvements to an existing situation. Implementing a change on the level described by these models, e.g., transitioning from a stage-gate product development system to an SBCE-based system, is likely to pose many practical challenges—especially in large organisations. Such models might thus be viewed as more indicative than directive of best practice; Blessing writes that prescriptive procedural models (including those on the meso-level) are seldom employed to direct DDP improvement (Blessing 1994).

5.2 Macro-level analytical models

Macro-level analytical models can be used to investigate and address the impact of a process' context. There are two main groups of such model: queueing models and system dynamics (SD) models. These are discussed in the next two subsections.

5.2.1 Queueing models

When a process is considered in context of the organisation that executes it, scarcity of resource and the need for workers to divide their effort among tasks from several sources often cause workload congestion and, consequently, delays. Queueing models provide a means to investigate and manage these macro-level issues.

The first group of models in this category incorporate dynamic simulations. For example, Adler et al. (1995) develop a queueing model to study workload and congestion effects in firms that handle multiple development projects concurrently. In such situations, even when a task has the information required to start, it must compete for attention with tasks from other projects. In their model, Adler et al. (1995) assume that projects can be grouped into different types, each of which is represented as an iterative task network that incorporates probability density functions to represent variation in individual project characteristics. The organisation is represented as a set of processing stations, each of which has a fixed capacity representing the number of individuals who can work on tasks of a certain type. In the simulation, projects are assumed to arrive at stochastic intervals, such that several are in progress at any time. The tasks from each project-in-progress are generated according to the precedence network for the corresponding project type, and queue for attention at the appropriate processing stations. Thus, the model captures the time which a project spends waiting for attention as well as the time spent performing tasks. Adler et al. (1995) argue that their model can accordingly predict cycle time more accurately than single-project approaches and, through what-if analysis, can provide useful insight into resourcing and congestion effects. Narahari et al. (1999) build on this work, arguing that similar insights can be gained through a simplified model in which each project is represented as a single job that flows through a network of processing stations, each representing a project stage. The stations are organised in a reentrant line to simulate loop-backs between stages. Although it does not represent complex concurrency within each project, this model can be used to assess project processing times and indicates how they can be improved through insights from queueing theory. In particular Narahari et al. (1999) recommend that workers prioritise jobs using policies designed to reduce variability, and that companies throttle the number of projects they take on concurrently.

Queueing models also often appear in engineering practice as the basis of tools that manage the queues of tasks in administrative processes such as the review and approval of design releases or of change orders. For example, this functionality is offered by commercial PLM solutions (Rangan et al. 2005). The emphasis is to automate the logistics of information flow and make people aware of tasks awaiting their attention, based on a process model which sets out the series of steps through which all jobs flow. Although providing useful infrastructure, these tools are in practice often associated with long process lead times which can cause many secondary problems, some of which are discussed by Oppenheim (2004). One approach to managing this is to provide visual depictions of the work-in-progress, through manually arranged or computerised visual management dashboards. Such dashboards usually show the sequence of process steps horizontally across the top of a computer screen or meeting room wall, and jobs awaiting attention are aligned underneath each step (Parry and Turner 2006). They may be discussed among a team on a regular basis with a view to managing priorities and bottlenecks. Other authors develop metrics for monitoring queueing in the DDP to enable and support continuous improvement (e.g., Beauregard et al. 2008).

Value stream mapping (VSM) is a workshop-based process mapping method that can help teams to identify bottlenecks, long lead times, unnecessary activity, and other wasteful situations in queueing processes, prior to understanding and addressing the root causes (Rother and Shook 2003). These problems commonly develop when responsibility for queueing processes is decomposed across departments or sites, such that no-one has overall responsibility for ensuring timely end-to-end flow. This often occurs for important back office processes in product development as well as in the production processes for which VSM was originally developed. Seeking to build on successes in this context, the VSM method has been adopted as the basis of the product development VSM (PDVSM) which is intended for the typically less-structured design processes. The PDVSM manual published by MIT Lean Aerospace Initiative states that 50–75% reduction in lead times of development processes can typically be expected when applying this method (McManus 2005). VSM is included in this section because it was originally developed to model and improve queueing systems in which inventory can accumulate between processing steps, although PDVSM arguably blurs the boundary between this idea and the task precedence models such as ASM that were discussed earlier.

Overall, queueing models are arguably most useful for handling relatively routine processes that can be perceived as workflows in which numerous jobs must follow the same sequence of operations. While these situations do frequently occur in development projects, the models may be less applicable to core design processes that involve less routineness.

5.2.2 *System dynamics models*

Another important contextual issue that is not emphasised in meso-level analytical models is the impact of influences and pressures on a process. System dynamics (SD) models address this issue by representing project governance structures and other influences, showing how these are coupled with the process and affect how it unfolds. Most such models draw on the work of Cooper (1980), who developed a canonical development project model which

can be adapted and calibrated for a particular situation. In Cooper's model, tasks or work packages are represented as interchangeable units which flow between four pools, as shown in Fig. 15. A project begins with all tasks in the original backlog of work and is considered complete once they have flowed through the system into work actually accomplished. The crux of the model as indicated in Fig. 15 is that quality problems in task execution cause a backlog of flawed work that needs to be redone, and some of this backlog may remain undiscovered for some time. Thus, the model shows why perceived progress tends to fall behind schedule, and why actual progress lags even further behind. This model structure became known as the rework cycle and has been adapted and extended to form the basis of many later SD models (Lyneis and Ford 2007). For instance, Ford and Sterman (1998) show how a sequence of rework cycles, each chained onto the next, can be used to represent overlapping stages of a development project. In these models, project influences are often dependent on the states of the activity pools and influence the rates at which tasks flow between pools. For example, one influence cycle may indicate that a rate of completing work (e.g., productivity, in Fig. 15) is influenced by schedule pressure, determined by whether the perceived amount of work remaining to be done is slipping behind a predetermined schedule. At the same time, increased work completion rate might reduce work quality causing progress to lag further behind perceptions. SD models can be useful to identify tipping points at which certain influences begin to dominate a situation. The equations that govern feedback effects are of great importance in determining a model's behaviour.

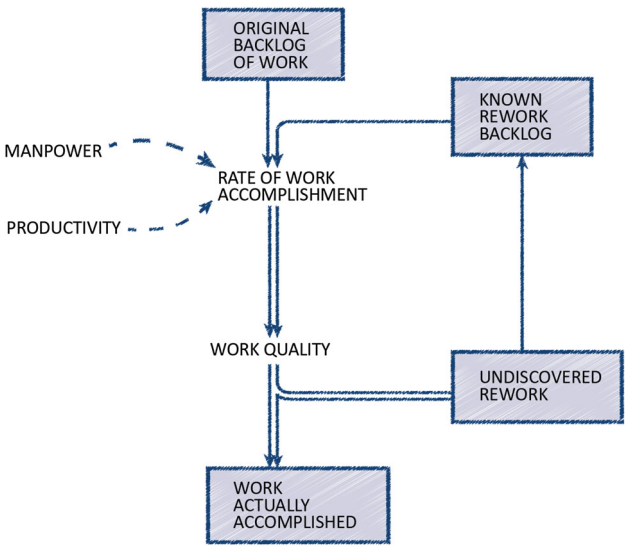

Fig. 15 Cooper's rework cycle (Cooper 1980). Reproduced with permission. Copyright, INFORMS, http://www.informs.org

A related macro-level analytical modelling approach is the use of qualitative causal networks to study project influences. This approach can be used to analyse factors that influence a DDP by modelling how they interact to exacerbate or suppress each other, and ultimately how these interactions might impact aspects of process performance. For example, Browning (1998) and Le (2013) both apply causal network modelling to analyse causes and effects of iteration in product development. They model the structure of influences relating to iteration by integrating individual factors and relationships revealed in case studies and prior research. Although the strengths of interactions might vary from one situation to the next, generic causal networks such as Fig. 16 may provide useful templates to guide the modelling and analysis of a specific situation (Le 2013).

For more information on models in this category, the reader is referred to Lyneis and Ford (2007) who provide a focused review of SD models applied to project management—many of which are either applicable or specific to the development project domain.

5.3 Macro-level abstract models

Abstract models of the DDP on the macro-level focus on clarifying its overall form and how design processes interact with their context. To recap, abstract models neither prescribe best practices as procedural models do nor provide concrete approaches for modelling a specific situation, as analytical models do.

One example of a model in this category is the capability maturity model for development or CMMI-DEV (CMMI 2010), which among other elements describes the system of process areas that exist in a generic development project or program (Table 2). Large organisations and their development projects may involve all these process areas, while smaller companies and projects may deal with many process areas in an ad-hoc way or not at all. The process areas can be categorised into core processes that directly create value, support processes, and management/control processes. Engineering design is mainly represented within 1 of the 22 process areas, although as a core value-adding process, it interacts strongly with the rest of the system. For instance, change in customer requirements influences the technical solution process, which in turn creates work for configuration management processes. This descriptive model, as summarised in Table 2, is helpful in the context of this article to indicate the relationship between the design process and the rest of a development project.

The diversity of models discussed thus far in the article makes it clear that many perspectives on the processes (and related systems) in an organisation are possible. The Zachman Framework (Zachman 1987) was one of the first

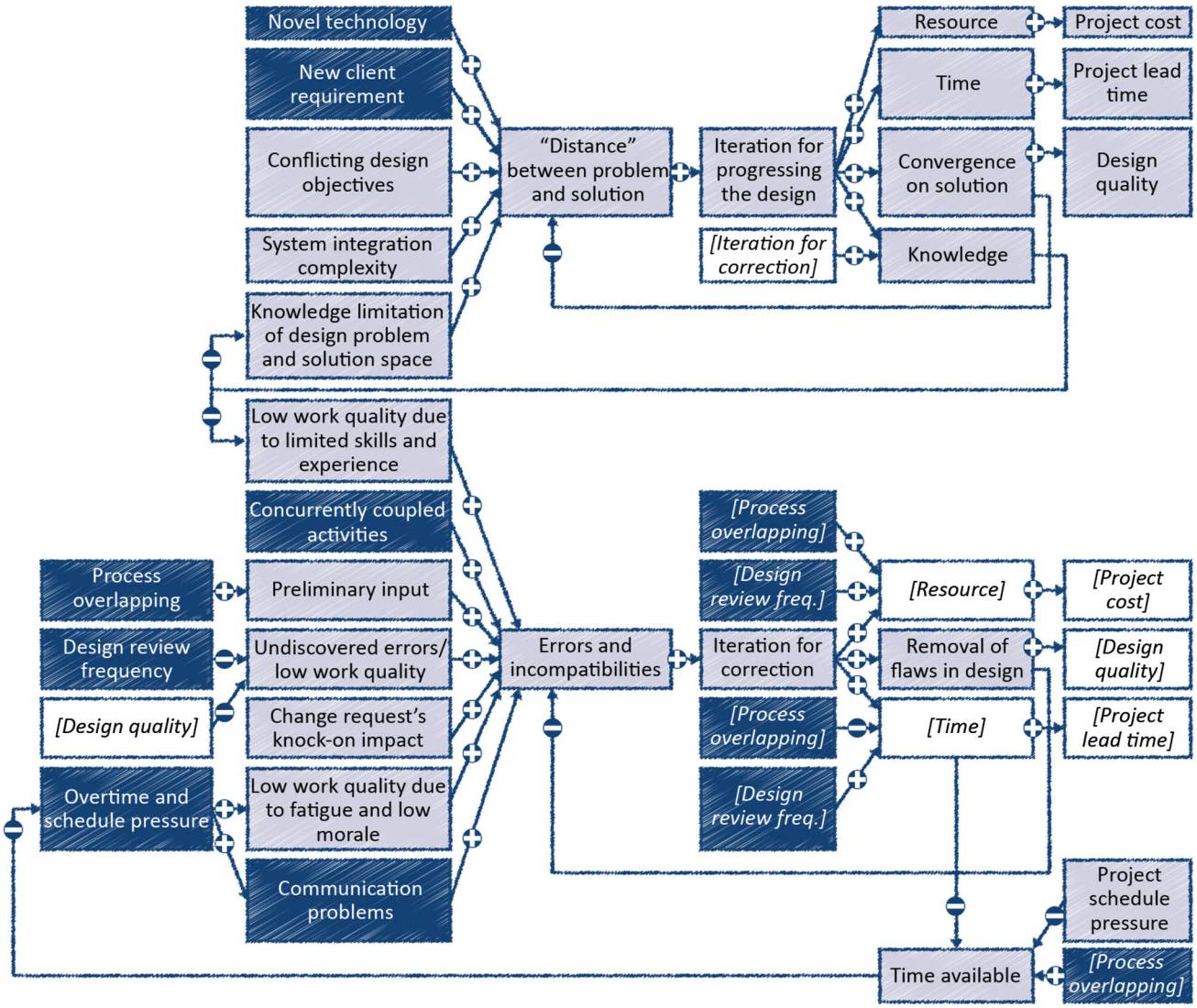

Fig. 16 Qualitative causal network integrating influences and effects relating to iteration in development projects. *Dark blue boxes* represent factors under management control. *Italic text* indicates that a factor appears at several points on the diagram. *Symbols on arrows* represent the reinforcing or suppressing nature of each influence. Figure reproduced from Le (2013) with permission of the author (color figure online)

models to provide a comprehensive picture of the possible perspectives. This kind of model is now known as an architecture framework. The Zachman Framework classifies perspectives on a two-dimensional grid. The first dimension indicates the question being asked by a model's users: What? How? Where? Who? When? and Why? The second dimension indicates the stakeholder asking the question: Planner, Owner, Designer, Builder, and Subcontractor. Zachman proposes that every modelling approach may be categorised as a combination of one question and one stakeholder, and that the alternative perspectives are "additive and complementary" (Zachman 1987). A more recent architecture framework is the US Department of Defense Architecture Framework (DoDAF) (DoD 2010). Architecture frameworks consider that different modelling approaches allow different views of a DDP system to be created. The different views must be integrated in the minds of their users. Browning (2009, 2014) argues for a centralised comprehensive DDP model from which customised views may be extracted according to each user's needs, recognising the practical difficulties of implementing such a system and keeping the information synchronised and up-to-date.

Other authors present more conceptual models. For example, the Integrated Product Engineering Model (iPeM) discussed by Albers and Braun (2011) combines a problem-solving cycle with a stage-based view of product development. This is framed as the so-called operation

Table 2 CMMI for Development v1.3 (CMMI 2010) includes definitions of 22 core development process areas. The engineering design process is part of Technical solution

CMMI-DEV process area	Purpose of processes in the area (summarised)
Causal analysis and resolution	Identify causes of selected outcomes and act to improve process performance
Configuration management	Establish/maintain configuration integrity of work products
Decision analysis and resolution	Analyse identified alternatives against established criteria to make decisions
Integrated project mgmt.	Define/execute integrated project processes tailored from standard processes
Measurement and analysis	Develop/sustain measurement capability for management information needs
Organisational process defn.	Establish/maintain process assets, and rules and guidelines for teams
Organisational process focus	Plan/implement/deploy process improvements considering current needs.
Org. performance mgmt.	Manage organisation performance proactively to meet business objectives
Org. process performance	Establish/maintain a quantitative approach to process performance.
Organisational training	Develop people, so they can perform their roles effectively and efficiently
Product integration	Ensure that the product is assembled from its components and behaves properly
Project monitoring and control	Understand project progress, so corrective actions can be taken when needed
Project planning	Establish and maintain plans that define project activities
Process and product quality	Objectively manage process and product compliance to standard
Quantitative project mgmt.	Achieve established quality and process performance objectives
Requirements development	Elicit, analyse and establish customer, product and component requirements
Requirements management	Manage requirements and ensure alignment with plans and work products
Risk management	Identify potential problems before they occur and mitigate adverse impacts
Supplier agreement mgmt.	Manage the acquisition of products and services from suppliers
Technical solution	Select, design, and implement solutions to requirements
Validation	Demonstrate that product's intended use is fulfilled in intended environment
Verification	Ensure that selected work products meet their specified requirements

system which transforms systems of objectives and requirements into systems of objects. The model is said to contain "the relevant elements to derive situation-specific PDP models" while "taking into account the dynamism and the uniqueness of product development processes" (Albers et al. 2016). Another example in this category is the Autogenetic Design Theory (ADT) of Vajna et al. (2005), which views design and development as a trial-and-error procedure guided by feedback due to situated selection pressures. This is said to occur at every level of product development. Vajna et al. (2005) present this as an evolutionary process, in particular as a cycle of mutation to generate alternatives, evaluation of alternatives, and selection according to situated pressures, followed by replication and recombination of successful candidates. They write that levels of complexity in the design increase as the evolutionary process proceeds, drawing an analogy to the outcomes of evolution by natural selection. Wynn et al. (2010) and Maier et al. (2014) develop a cybernetic model of the DDP, arguing that the process participants interact through consideration of an ecosystem of models, including representations of the emerging design as well as DDP models. The models are said to mediate dynamic interactions among individual process participants and their design contexts. This perspective is used to identify eight factors that influence the effectiveness of models and modelling in guiding a DDP towards desired outcomes in the presence of uncertainty, disturbance, and situated decisions. Siyam et al. (2015) present a Value Cycle Model which presents complex product development as a network of roles related to the process of defining, creating, and delivering value with respect to stakeholders involved in product development. This model is used to position tools and approaches that may be used to improve the DDP from a value perspective. Pich et al. (2002) present a formal conceptual model that characterises projects according to information adequacy with a view to choosing an appropriate management strategy. They consider three such strategies, which correspond to models discussed earlier in this article. First, instructionism involves programming activities and perhaps contingency plans in detail, as per PERT/GERT and similar approaches. Pich et al. (2002) argue that this is suitable only if information about the situation and about the effect of actions is deemed adequate. In situations with many unknown unknowns, a strategy involving learning (i.e., deliberately iterative, experimental approaches such as DPD, Agile, and IID) and/or selectionism (i.e., pursuing multiple alternatives in

parallel until there is enough information to choose between them, as per SBCE) may be more effective.

In summary, abstract models like these can be useful to frame analyses of the DDP on the macro-level. However, they are rather conceptual in nature and may require significant insight and interpretation to apply.

5.4 Macro-level MS/OR models

The final category of models concerns computational or mathematical studies of factors governing processes on the macro-level.

The first group of models in this category consider the overlapping of two consecutive project stages or tasks. These models are classified as macro-level because they focus on managerial decisions without representing the numerous tasks in a process flow. Much work on this topic was inspired by Krishnan et al. (1997) who study how preliminary transfer of information from an upstream stage, such as product design, allows a downstream stage, such as production design, to be started early. Because it is only an estimate of the final value, the preliminary information will be subject to one or more updates, each of which causes downstream rework (Fig. 17). This is modelled as a curve that defines the evolution of the upstream task's output towards a final value, and another defining how the sensitivity of the downstream task to changes increases over time. Krishnan et al. (1997) develop optimal overlapping strategies considering the forms of the two curves. Loch and Terwiesch (1998) further analyse the two-stage overlapping situation, focusing on the communication that enables overlapping. Their model considers that holding

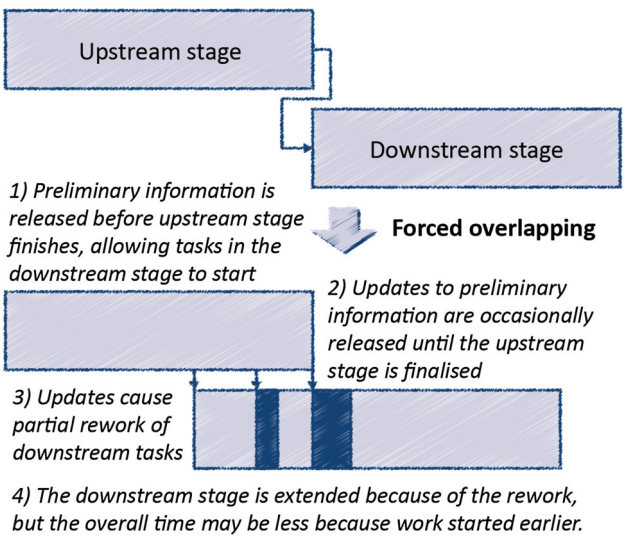

Fig. 17 Effects of overlapping DDP stages. Illustrates concepts discussed by Krishnan et al. (1997)

meetings to communicate more frequently during the overlapping period reduces iteration impact, because each change released by the upstream task will require more work to be redone the later it is dealt with, since more of the dependent work will be completed. However, meetings also require time. Optimal policies for overlapping are derived algebraically under these assumptions. Joglekar et al. (2001) assume that each of the two overlapping tasks generates 'design performance' at a fixed rate while also reducing the performance generated by its partner, causing rework to regain the prior level. They use algebraic manipulations to show how the relative rates of performance generation and the coupling strength between the tasks determine the optimal overlap. Again focusing on two tasks, Roemer and Ahmadi (2004) investigate the relationship between overlapping and crashing, i.e., increasing work intensity to reduce duration while increasing effort. They conclude that these approaches should be considered together and that the intensity of work should follow a certain pattern to minimise the rework caused by overlapping. The models described above incorporate many simplifying assumptions that assist with manipulating the algebra. Other researchers study similar issues using Monte Carlo simulation which allows study of more complex problems involving more factors and variables. For instance, the model developed by Bhuiyan et al. (2004) focuses on how sequentially dependent process phases can be overlapped to reduce development time at the risk of causing iteration at the phase exit review. They show that this risk can be mitigated by increasing the degree of functional interaction between engineering functions within each phase, although this causes more iteration within the phases.

Second, some researchers take an MS/OR approach to analyse the situations in which different macro-level process structures are appropriate. For instance, Bhattacharya et al. (1998) study to what degree a flexible process in which a design specification is evolved by repeated user feedback can be justified, considering that this may increase product attractiveness and thus sales, but leaves less time to optimise the design which may result in higher production costs. Several factors that should influence the choice of process structure are studied, including market uncertainty, the firm's appetite for risk, and the value of information that can be gained from customer feedback. Loch et al. (2001) consider when testing of design alternatives should be done in parallel (as per SBCE) allowing quick convergence to a solution, or sequentially, which allows for learning from each test to inform the next in a process of iterative improvement. Their model shows that parallel testing is most useful if the cost of tests is low or the time required to complete each test is significant, and if the tests are effective in revealing information about the

designs. Suss and Thomson (2012) develop a discrete-event simulation model called the Collaborative Process Model (CoPM) that represents an engineering design process on three levels: a stage-gate structure; the activities and their interdependencies within each stage; and the actors or teams that carry out the activities. Among other insights, Suss and Thomson (2012) use their model to show that Scrum (an IID approach in which each iteration involves a short period of intense communication followed by a design review) is more effective than a traditional staged process in cases of high uncertainty within the process.

6 Discussion

6.1 Recap and summary of the DDP models

Sections 3, 4, 5 highlight that models of the design and development process span a vast range of issues and perspectives. Work in the abstract and MS/OR categories examines the DDP on a relatively conceptual level. The foci of models in these categories range from the individual designer's problem-solving processes through to macro-scale project processes. Although they offer useful insights which can help to guide process improvement activities, such models are usually too general to provide detailed, implementation-level advice (and, we think, this is usually not intended by the respective researchers).

On the other hand, approaches in the procedural and analytical categories aim to directly support improvements to the design and development process. Significant differences in philosophies and modelling assumptions are apparent across these categories. In common with the abstract and MS/OR approaches, none of the models or even categories of models are agreed to adequately represent all aspects of the DDP. Thus, the modeller must select an appropriate approach for the context at hand. It is hoped that by providing an overview of the models and commenting on their advantages and limitations, the present article may facilitate this task.

6.2 Relationships across the framework categories

In this article, we have chosen to organise DDP models primarily according to their scope. This reflects the main clustering of approaches in the literature, in the sense that many articles' bibliographies concentrate on work within one of the three levels shown in Fig. 1. However, this is not the only possible organisation and interdependencies do exist between these levels. Micro-level models can provide insight relevant to the meso-level, for instance, because rework in meso-level processes is ultimately driven by design decisions made by individuals—even though those decisions' effects may unfold over a long timescale if many individuals and/or departments are involved. Similarly, meso-level models provide insight into macro-level process characteristics. For example, the patterns of information flow between two departments such as design and test will determine the level of overlapping that might be appropriate between those departments, and whether a rigid stage-gate model would be appropriate. Analyses that cross the levels as we defined them seem to be relatively rare at present. We suggest that teasing out links between the levels could be a useful direction for further work. To give just one example, insights from research into design negotiation might present opportunities to improve the probabilistic assumptions underlying treatments of iteration in some meso-level analytical models.

6.3 DDP characteristics and implications for models

In the introduction to this article, a number of important characteristics of the design and development process were mentioned, in particular its iteration, novelty, and complexity. These are now revisited to consider how the models treat them and to identify implications for further research.

6.3.1 Iteration

The iterative nature of design and development features prominently in almost all the models reviewed. Wynn and Eckert (2017) argue that there are many perspectives on iteration and find that most approaches only emphasise a few "stereotypes". The significance for the present article is that DDP models tend to idealise complex iterative situations in a way that focuses attention on a few selected issues. For example, the spiral model developed by Evans (1959, Fig. 5) implies that iteration helps to converge on a design, and may be desirable, while the Q-GERT model of Taylor and Moore (1980, Sect. 4.2.1) indicates that iteration is mainly caused when tasks reveal problems, and thus is undesirable. Because each stereotype suggests a quite different perspective on the causes, effects, and behaviours of iteration, it is important to understand the iterative characteristics of a real-world situation and select a model that focuses on the appropriate stereotype(s) (Wynn and Eckert 2017). A model that is poorly matched to the iterative situation may not yield much insight and may draw the focus of attention away from pertinent issues. Selecting an appropriate model may be difficult if the modeller is not aware of the range of approaches that are available; the present article may be informative in this regard. Opportunities for further work on this topic include developing methods to assess the iterative characteristics of real-world

situations to match them to appropriate models, and developing hybrid models that blend or nest the stereotypes.

Considering analytical and MS/OR approaches in particular, we believe that some quite common assumptions regarding iteration deserve further attention. First, many methods require a modeller to specify activities, decision points, or dependencies that can cause iteration to be triggered. The choice of which triggers to include in a model will be influenced by the practicalities of modelling, and iteration may often appear in other places, or may appear to be outside the level of detail of the model (Browning 1998). One area for further work is to develop methods to assess how iteration is triggered and which triggers are important to incorporate in a process model. As well as indicating where iteration may occur, many approaches incorporate a mathematical model of when it is triggered. This is most commonly stochastic, but the constant and independent probabilities often used may not be a good model of how iteration occurs in practice (Smith and Tjandra 1998). One contributing factor is that choices are available regarding how to manage iterations (Wynn 2007). For example, companies may accept some problems so they can release a design on time, with the intention to work out those issues during production or after the design is in service. Exploring the most effective ways to model iteration initiation is, therefore, another area for further work. Finally, as noted earlier, some simulation schemes can suffer from logical issues related to iteration, such as deadlock, if a model is not carefully formulated (Karniel and Reich 2009). For all these reasons, it remains difficult to adequately represent iterations in practice especially in unstructured or nonroutine processes. We suggest that there are opportunities for further work on how DDP models can be used to support practice despite their limited fidelity, for instance, using them in ways that recognise their limitations (e.g., Kerley et al. 2011).

6.3.2 *Novelty*

Another challenge faced when developing models of the DDP is that every project and design situation is in some respect unique, or at least unique to its participants. Different models deal with this challenge in different ways. Abstract, procedural, and MS/OR models describe the DDP in a generic way expected to be valid in many different contexts. However, as noted earlier, this may present difficulties for practical application. Many analytical models are based on the principle that although each DDP is different, there is an underlying process architecture within each company that remains essentially constant from one product to the next, and that may be modelled. Thus, a process model based on past experience may help to derive insights for future DDPs in similar contexts. However, it seems not entirely clear how process similarity should be understood or assessed, nor what its implications for modelling and analysis might be. In practice, processes change over time, for instance as new technologies become available and become integrated into the designs and as new software tools are rolled out. In the development of complex products such as aircraft, this change can occur on a timescale that is significant relative to the project timescale. Because of this, further research to explore how models can be most effective taking into account an evolving process might prove to be useful. One possibility is to map DDP models or model fragments to characteristics of the situations in which they are valid, such that a process can be progressively instantiated as design decisions are made and/or can be adapted to a particular context (e.g., Chung et al. 2002; Muller et al. 2007).

6.3.3 *Complexity*

We have already discussed insights into DDP complexity revealed through several models. These include insights on the interrelationships between the design process, the properties of the emerging design, and the context into which that design will be delivered (e.g., Gero and Kannengiesser 2004); insights on the information flows that emerge between participants as they coordinate their response to inevitable unplanned events (e.g., Cohen 1992); insights on the impact of structural complexity in task networks on design iteration and convergence (e.g., Braha and Bar-Yam 2007); and insights on the dynamic complexity caused by multiple intertwined influence loops as participants guide a project towards desirable outcomes (e.g., Lyneis and Ford 2007). As well as generating insights, models can help to manage DDP complexity by presenting selected issues in a simplified way. At the same time, when working with a model, attention is focused on the issues that are emphasised. Knowledge of the full range of models available is important not only to select an appropriate approach for a given situation as suggested in Sect. 6.3.1, but also to ensure awareness of how different models might influence perceptions of the process (Wynn 2007).

Another issue relating to model complexity is that a modeller must choose the scope and granularity of their representation (Maier et al. 2017). This inevitably involves simplifications, and the consequences may be especially important to consider when seeking insights from mathematical or computational models. For example, Kerley et al. (2011) argue that a DDP simulation model should not be viewed as an attempt to create a "perfect simulacrum", but as a tool for "providing enough information to the stakeholders to facilitate debate and support them in

making evidence-based judgements about the feasibility and consequences of implementing the suggested changes" (Kerley et al. 2011). A related consideration is whether results and insights from numerical analysis can be expected to converge as the level of detail in the model is increased, or as the severity of simplifying assumptions is reduced. For example, would a Task DSM clustering algorithm yield the same insights if the same process were modelled at different levels of abstraction, or by different people? Future work to develop more systematic guidelines to make appropriate choices while modelling might prove useful to practitioners (Gericke et al. 2016).

A third set of issues relating to complexity in the context of analytical approaches concerns the practical constraints on constructing large models. Some authors have proposed that these problems of scale can be addressed by developing process libraries from which case-specific models can be more quickly assembled (Austin et al. 1999; Park and Cutkosky 1999; Wynn et al. 2006). This appears to be a promising approach for situations which can be decomposed hierarchically and in which the subprocess contents are relatively routine. For example, the process library in the ADePT method was found to account for more than 90% of the activities required in application case studies (Austin et al. 1999). It should be noted that this work focused on the construction sector, not engineering design. Appropriate software support might also help to manage large and complex models, for instance by generating filtered views customised to the needs of each user. However, these specialised tools are often not available in practice (Eckert et al. 2017).

6.4 Process models in DDP practice

Research literature can sometimes seem to present a rather theoretical view of models which may not fully reflect how they are used in practice. In reality, companies do not use any one model or modelling approach exclusively. Many fragmentary models coexist in a company and their contents can overlap to varying degrees (Eckert et al. 2017). Models vary in terms of the approach or notation used, the scope and level of detail, and the level of fidelity. There is often no organised framework in which most models used within a company are positioned, and if such a framework exists, it may not be appropriately utilised by everyone. Browning (2002) views this fragmentation of process models as "extremely undesirable", suggesting that it may contribute to difficulties in organising and coordinating a process, with the consequence that information may not flow to the right people in timely fashion.

Although undesirable, this situation is, for now, usually the reality. People need to consider multiple DDP models to find information about their processes, or may gain that information by asking their colleagues and perhaps building their own models. The value of a model often lies in helping people to frame and analyse a complex situation—models must be interpreted by bringing them together with knowledge of the application context, and simulated in the minds of their users to understand their implications and guide decisions (Andreasen et al. 2015). There is an opportunity for further research to examine the properties of this system of interactions between models and their stakeholders in a company, and how those properties might affect the coordination and performance of the DDP.

The different types of model are used in different ways. Procedural models are typically evolved in companies to meet their specific needs (Tomiyama et al. 2009). In particular, most firms customise the stage-gate model to their processes and the customised version will be familiar to most employees, although in a multi-year program, it may not provide much guidance for day-to-day activity. Other procedural models such as the PDCA cycle are typically associated with particular improvement initiatives in a company, and depending on the success of the particular initiative might be accepted to a greater or lesser degree. The main value of these models in practice is arguably to assist in communicating methodological insights to a large number of employees, and as such, clarity of exposition may be one of their most important characteristics.

In terms of analytical models, many large companies have developed a set of process maps which, in some sectors such as aerospace and automotive, are required by regulatory authorities to demonstrate that the company can explain how its products are developed and show that required process steps such as validation activities are appropriately performed (Browning 2002). The effort to keep this information up-to-date in the face of changing processes and technology can be significant and practice can deviate substantially from what these models portray. Other analytical process models used in practice are developed as an early step in the process design or improvement initiatives to generate understanding of the process in focus. Such models, often using notations such as BPMN, are essentially isolated, because the initiatives that generate them are often very limited in scope. As a result, they may not continue to deliver benefit once those initiatives are finished. Some analytical models such as task network simulation, system dynamics, and DSM may find limited application in a company but are often limited to trials driven by the personal interest of individuals, while others such as agent-based models and rule-based models remain mostly in the research domain.

Finally, abstract and MS/OR models are arguably not intended for direct application in industry and are probably

not often used in that context, although the insights developed from them may be of value to practitioners.

6.5 Some challenges of DDP modelling in practice

Whether it concerns a high-level procedural model or a detailed analytical model, for a process modelling initiative to impact beyond a few specialists, the models should be easy to understand and deliver clear benefit. Even if practitioners might in principle derive benefits from models and modelling, in an industry context, there are many pressures competing for time and attention. During product development, modelling and improving processes are often seen as non-critical activities and delivery of the next program often takes priority. Process improvement and its related modelling activities are seen by many design personnel as tasks that can be left for later life-cycle phases, for example for improving production processes when ramping up production. Another issue is that development projects can often seem quite ad-hoc, with much attention devoted to chasing for information and attention, and addressing issues and problems as they emerge. Thus, from a practitioner's perspective, many DDP models can seem idealised and sterile and not relevant to the day-to-day activities of the design engineers who must participate in developing or implementing them. Due to the difficulties of bringing such personnel on board and the limited available time, modellers may often choose the 'low-hanging fruit' and focus their efforts on support processes such as engineering change management which have a more repeatable nature and often involve administrative instead of technical issues.

Another issue of great importance to practitioners is the availability of tools for modelling. Large companies often prescribe tools and process modelling notations to standardise the information that is generated. The benefits of this approach include facilitating training, understanding, and curation of models—but it also forces modellers to work within a particular tool and notation that may not be suitable for every purpose. The approach that is chosen is often one of the main task precedence representations such as BPMN or EPC which are mainly oriented towards business process modelling and, arguably, are not ideal for the DDP context due to its iteration, novelty, and complexity. Many research approaches that might better address these issues are not implemented in deployable tools at all, and those that are both implemented and available for download or purchase must compete against the offerings of large established software suppliers. Finally, DDP modelling and improvement requires an understanding of engineering issues alongside skills such as workshop facilitation and change management. This is challenging work, but it is often perceived in companies as non-critical, so it may be difficult to attract and retain personnel with the ideal skill set.

6.6 Relationship of this article to earlier reviews

Our intention to contribute an integrating narrative, combined with the space constraints of a journal article, led us to focus on key publications rather than attempting a complete listing of all work that relates to each framework category. Throughout the text, we have provided pointers to other literature reviews that provide focused analyses of particular topics.

To demonstrate the relationship of this article to earlier reviews, 30 useful reviews were identified and mapped against the 12 categories of the framework depicted in Fig. 1. The result, as shown in Table 3, demonstrates that almost all reviews which we identified focus on a small subset of the categories considered here. Although many of these reviews offer comprehensive and insightful analyses within their scope, the table shows that no prior article maps the overall topology of the literature as done here.

More specifically, we found only three prior reviews that cover more than 50% of the categories considered here. In the first of the three, Eder and Weber (2006) focus on comparing procedural and abstract models to the work of Hubka, and do not cover the analytical or MS/OR categories (with very few exceptions). The second comprehensive review, published by Browning and Ramasesh (2007), offers thorough coverage and analysis of process models in product development and project management, but contributions from design research are almost entirely out-of-scope. Finally, Wynn (2007) discusses models in 10 of the 12 categories, but does not consider the substantial contributions made by MS/OR models (with a single exception). In addition, it may be noted that research in this area has substantially developed in the years since these reviews were published.

Overall, Table 3 provides a starting point for further reading on specific topics, and also confirms that earlier reviews each cover only a subset of the categories that we identified. The present article has been written to address this gap. It is hoped that our framework will provide a useful integrating overview of the key ideas and will help to articulate the value of individual DDP models considering the broad landscape of research in the area.

7 Concluding remarks

Process models and modelling approaches have been created to address many different issues in the DDP. The organising framework developed in this article, summarised in Fig. 1, highlights the value of models and

Table 3 Thirty selected publications that incorporate useful reviews of design and development process models

	Micro-level coverage				Meso-level coverage				Macro-level coverage				Σ
	Pr	An	Ab	MS	Pr	An	Ab	MS	Pr	An	Ab	MS	
Finger and Dixon (1989)	✓		✓		✓								3
Roozenburg and Cross (1991)	✓		✓		✓								3
Cross and Roozenburg (1992)	✓		✓		✓								3
Konda et al. (1992)	✓	✓	✓		✓				✓				5
Cross (1993)	✓		✓		✓		✓						4
Bahrami and Dagli (1993)	✓		✓										2
Blessing (1994)	✓		✓		✓	✓	✓		✓				6
Evbuomwan et al. (1996)	✓		✓		✓				✓				4
Smith and Morrow (1999)						✓		✓		✓		✓	4
Dubberly (2004)	✓		✓		✓				✓				4
Wynn and Clarkson (2005)	✓		✓		✓		✓		✓				5
O'Donovan et al. (2005)		✓				✓							2
Eder and Weber (2006)	✓	✓	✓		✓		✓		✓		✓		7
Wynn (2007)	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		10
Browning and Ramasesh (2007)	✓	✓				✓		✓	✓	✓	✓	✓	8
Lyneis and Ford (2007)										✓		✓	2
Howard et al. (2008)	✓		✓		✓				✓				4
Tomiyama et al. (2009)	✓		✓		✓	✓	✓		✓				6
Sharafi et al. (2010)	✓				✓				✓				3
Gericke and Blessing (2011)	✓		✓		✓				✓				4
Gericke and Blessing (2012)	✓		✓		✓				✓				4
Amigo et al. (2013)		✓				✓				✓			3
Mohd Saad et al. (2013)	✓												1
Andreasen et al. (2015)	✓		✓		✓		✓		✓		✓		6
Costa et al. (2015)					✓				✓		✓		3
Chakrabarti and Blessing (2015)	✓	✓	✓		✓		✓		✓				6
Browning (2016)						✓		✓					2
Bobbe et al. (2016)					✓				✓				2
Wynn and Eckert (2017)			✓					✓				✓	3
Eckert et al. (2017)		✓				✓					✓		3

The table provides a starting point for further reading. It also demonstrates that previously published reviews cover only subsets of the 12 model categories considered in this article. A tick indicates that one or models from a category are reviewed in a publication

Pr procedural, *An* analytical, *Ab* abstract, *MS* MS/OR. These categories are defined in Table 1

modelling in accentuating different aspects of the DDP and maps the topology of the literature. Both research and practice suggest that most situations may be usefully described by more than one category of model. At the same time, a model can provide insight on different levels depending on how it is interpreted and applied. Each model emphasises different elements from a web of interconnected ideas, offering different terminology and different visual depictions. In many cases, the perspectives can be difficult to reconcile. We concur with Bahrami and Dagli (1993) and others in recommending a pluralistic approach, in which the DDP is simultaneously perceived from many points of view, from the individual designer's problem-solving process through to the need for continuous improvement. At the same time, it should be recognised that some models represent conflicting philosophies. A design and development process should be designed considering the requirements and constraints of its context (Kolberg et al. 2014), and thus, not all models will be relevant to every situation.

Many questions remain open to debate. Although a fully integrated perspective on the design and development process might be difficult to attain, we suggest that there are numerous opportunities to selectively synthesise insights across layers and categories of our framework. Overall, it is hoped that the framework and review presented in this article may prove useful to researchers seeking to position their work, as well as to educators and

practitioners seeking an overview of the approaches and perspectives that have been developed.

Acknowledgements The authors gratefully acknowledge past and present collaborators, including Claudia Eckert, Martin Stacey, and Vince Thomson, for many discussions on DDP models. Some material in this article was adapted and substantially extended from earlier work in Wynn and Clarkson (2005), Wynn (2007), and Wynn and Eckert (2017). We also thank the Editor, and the anonymous reviewers for sharing their insights. Figure 4 is reprinted from Design Studies, vol. 25, JS Gero, and U Kannengiesser, The Situated Function-Behaviour-Structure Framework, Pages 373-391, Copyright 2004, with permission from Elsevier. Figure 6 is reproduced from Conceptual Design for Engineers (3rd Edition), Ch. 1: Introduction, 1999, Page 2, MJ French, ©Springer-Verlag Berlin Heidelberg 1999. With permission of Springer. Figure 7 is reproduced from Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge, Ch. 8 Design Science for TS-Types, 1999, page 197, V Hubka and WE Eder, ©Springer-Verlag London Limited 1996. With permission of Springer. Figure 10 is reproduced from Research in Engineering Design, A Model-Based Method for Organizing Tasks in Product Development, Volume 6, 1994, page 3, SD Eppinger, DE Whitney, RP Smith and DA Gebala, ©1994 Springer-Verlag London Limited. With permission of Springer. Figure 13 is reprinted from Business Horizons, Vol. 33, RG Cooper, Stage-gate systems: a new tool for managing new products, Pages 44–54, Copyright 1980, with permission from Elsevier.

References

Adler PS, Mandelbaum A, Nguyen V, Schwerer E (1995) From project to process management: an empirically-based framework for analyzing product development time. Manag Sci 41(3):458–484

Agogué M, Kazakçi A (2014) 10 years of C-K theory: a survey on the academic and industrial impacts of a design theory. In: Chakrabarti A, Blessing MLT (eds) An anthology of theories and models of design: philosophy, approaches and empirical explorations. Springer-Verlag, London, pp 219–235

Ahmadi R, Wang R (1999) Managing development risk in product design processes. Oper Res 47(2):235–246

AitSahlia F, Johnson E, Will P (1995) Is concurrent engineering always a sensible proposition? IEEE Trans Eng Manag 42(2):166–170

Al-Ashaab A, Golob M, Attia UM, Khan M, Parsons J, Andino A, Perez A, Guzman P, Onecha A, Kesavamoorthy S, Martinez G, Shehab E, Berkes A, Haque B, Soril M, Sopelana A (2013) The transformation of product development process into lean environment using set-based concurrent engineering: A case study from an aerospace industry. Concurr Eng 21(4):268–285

Albers A, Braun A (2011) A generalised framework to compass and to support complex product engineering processes. Int J Prod Dev 15(1–3):6–25

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

Altshuller G (1999) The innovation algorithm: TRIZ, systematic innovation and technical creativity. Technical Innovation Center, Inc., Worcester

Amigo CR, Iritani DR, Rozenfeld H, Ometto A (2013) Product development process modeling: state of the art and classification. In: Abramovici M, Stark R (eds) Smart product engineering: proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11th–13th, 2013. Springer, Berlin Heidelberg, pp 169–179

Andreasen MM (1980) Machine design methods based on a systematic approach—contribution to a design theory. Dissertation, Department of Machine Design, Lund University, Sweden **(in Danish)**

Andreasen MM (2011) 45 years with design methodology. J Eng Des 22(5):293–332

Andreasen MM, Hein L (2000) Integrated product development. IPU, Institute for Product Development, Technical University of Denmark, Lyngby/Copenhagen

Andreasen MM, Hansen CT, Cash P (2015) Conceptual design: interpretations, mindset and models. Springer International Publishing, Cham, Switzerland

Antonsson E, Otto K (1995) Imprecision in engineering design. J Mech Des 117:25–32

Archer LB (1965) Systematic method for designers. Council of Industrial Design, London

Asimow M (1962) Introduction to design. Prentice Hall, Englewood Cliffs, NJ

Aurisicchio M, Bracewell R (2013) Capturing an integrated design information space with a diagram-based approach. J Eng Des 24(6):397–428

Austin S, Baldwin A, Li B, Waskett P (1999) Analytical design planning technique: a model of the detailed building design process. Des Stud 20(3):279–296

Austin S, Baldwin A, Li B, Waskett P (2000) Analytical design planning technique (ADePT): a dependency structure matrix tool to schedule the building design process. Constr Manag Econ 18(2):173–182

Bahrami A, Dagli CH (1993) Models of design processes. In: Sullivan WG, Parsaei HR (eds) Concurrent engineering, contemporary issues and modern design tools. Springer, Dordrecht, pp 113–126

Bartolomei JE, Hastings DE, de Neufville R, Rhodes DH (2012) Engineering systems multiple-domain matrix: An organizing framework for modeling large-scale complex systems. Syst Eng 15(1):41–61

Baxter D, Gao J, Case K, Harding J, Young B, Cochrane S, Dani S (2007) An engineering design knowledge reuse methodology using process modelling. Res Eng Des 18(1):37–48

Beauregard Y, Thomson V, Bhuiyan N (2008) Lean engineering logistics: load leveling of design jobs with capacity considerations. Can Aeronaut Sp J 54(2):19–30

Bhattacharya S, Krishnan V, Mahajan V (1998) Managing new product definition in highly dynamic environments. Manag Sci 44(11):S50–S64

Bhuiyan N, Gerwin D, Thomson V (2004) Simulation of the new product development process for performance improvement. Manag Sci 50(12):1690–1703

Blessing LTM (1994) A process-based approach to computer-supported engineering design. University of Twente, Enschede

Bobbe T, Kryzwinski J, Woelfel C (2016) A comparison of design process models from academic theory and industry practice. In: Marjanović D, Štorga M, Pavković N, Bojčetić N, Škec S (eds) Proceedings of DESIGN 2016, the 14th International Design Conference, Dubrovnik, Croatia, May 16–19, Design Society, pp 1205–1214

Boehm BW (1988) A spiral model of software development and enhancement. Computer 21(5):61–72

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 (1998a) A mathematical theory of design: foundations, algorithms and applications. Springer Science & Business Media, Dordrecht

Braha D, Maimon O (1998b) The measurement of a design structural and functional complexity. IEEE Trans Syst Man Cybern Part A Syst Hum 28(4):527–535

Braha D, Reich Y (2003) Topological structures for modeling engineering design processes. Res Eng Des 14(4):185–199

Browning TR (1998) Modeling and analyzing cost, schedule and performance in complex system product development. PhD thesis, MIT

Browning TR (2002) Process integration using the design structure matrix. Syst Eng 5(3):180–193

Browning TR (2009) The many views of a process: toward a process architecture framework for product development processes. Syst Eng 12(1):69–90

Browning TR (2014) Managing complex project process models with a process architecture framework. Int J Proj Manag 32(2):229–241

Browning TR (2016) Design structure matrix extensions and innovations: a survey and new opportunities. IEEE Trans Eng Manag 63(1):27–52

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

Browning TR, Ramasesh RV (2007) A survey of activity network-based process models for managing product development projects. Prod Oper Manag 16(2):217–240

Browning TR, Fricke E, Negele H (2006) Key concepts in modeling product development processes. Syst Eng 9(2):104–128

Buur J (1990) A theoretical approach to mechatronics design. PhD dissertation, Technical University of Denmark

Chakrabarti A, Blessing L (2015) A review of theories and models of design. J Indian Inst Sci 95(4):325–340

Chakrabarti A, Sarkar P, Leelavathamma B, Nataraju B (2005) A functional representation for aiding biomimetic and artificial inspiration of new ideas. Artif Intell Eng Des Anal Manuf 19:113–132

Chandrasekaran B (1990) Design problem solving: a task analysis. AI Mag 11(4):59

Cho SH, Eppinger SD (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, Stanford University

Chung MJ, Kwon P, Pentland BT (2002) Making process visible: a grammatical approach to managing design processes. J Mech Des 124(3):364–374

Clarkson PJ, Hamilton JR (2000) 'Signposting', a parameter-driven task-based model of the design process. Res Eng Des 12(1):18–38

Clarkson PJ, Melo A, Eckert C (2000) Visualization of routes in design process planning. In: Banissi E, Bannatyne M, Chen C, Khosrowshahi F, Sarfraz M, Ursyn A (eds) Proceedings of the 2000 IEEE International Conference on Information Visualization, London, England, Jul 19–21, IEEE Computer Society, pp 155–164

CMMI (2010) CMMI for development, version 1.3. Tech. rep., Carnegie-Mellon University

Cohen GP (1992) The virtual design team: an information-processing model of design team management. PhD thesis, Stanford University

Colquhoun GJ, Baines RW, Crossley R (1993) A state of the art review of IDEF0. Int J Comput Integr Manuf 6(4):252–264

Cooper KG (1980) Naval ship production: a claim settled and a framework built. Interfaces 10(6):20–36

Cooper RG (1990) Stage-gate systems: a new tool for managing new products. Bus Horiz 33(3):44–54

Costa DG, Macul VC, Costa JMH, Exner K, Pförtner A, Stark R, Rozenfeld H (2015) Towards the next generation of design process models: A gap analysis of existing models. In: Weber C, Husung S, Cantamessa M, Cascini G, Marjanović D, Venkataraman S (eds) Proceedings of the 20th International Conference on Engineering Design (ICED 15), Milan, Italy, Jul 27–30, Design Society, vol 2, pp 441–450

Cross N (1993) Science and design methodology: a review. Res Eng Des 5(2):63–69

Cross N, Roozenburg N (1992) Modelling the design process in engineering and in architecture. J Eng Des 3(4):325–337

Crowder RM, Robinson M, Hughes HP, Sim YW (2012) The development of an agent-based modeling framework for simulating engineering team work. IEEE Trans Syst Man Cybern Part A Syst Hum 42(6):1425–1439

Cusumano MA, Selby R (1997) How Microsoft builds software. Commun ACM 40(6):53–61

Danesh MR, Jin Y (2001) An agent-based decision network for concurrent engineering design. Concurr Eng 9(1):37–47

Danilovic M, Browning TR (2007) Managing complex product development projects with design structure matrices and domain mapping matrices. Int J Proj Manag 25(3):300–314

DoD (2010) DoDAF architecture framework version 2.02. Tech. rep., US Department of Defense

Dori D (2002) Object-Process Methodology: a holistic systems paradigm. Springer-Verlag, Berlin, Heidelberg

Dorst K, Cross N (2001) Creativity in the design process: co-evolution of problem-solution. Des Stud 22(5):425–437

Dubberly H (2004) How do you design? A compendium of models. Dubberly Design Office, San Francisco

Dym CL, Little P, Orwin EJ (2014) Engineering design: a project-based introduction, 4th edn. Wiley, New York

Eckert CM, Clarkson PJ (2010) Planning development processes for complex products. Res Eng Des 21(3):153–171

Eckert CM, Wynn DC, Maier JF, Albers A, Bursac N, Xin Chen HL, Clarkson PJ, Gericke K, Gladysz B, Shapiro D (2017) On the integration of product and process models in engineering design. Des Sci 3(3):1–41

Eder WE (2011) Engineering design science and theory of technical systems: legacy of Vladimir Hubka. J Eng Des 22(5):361–385

Eder WE, Weber C (2006) Comparisons of design theories. In: AEDS 2006 Workshop, Oct 27–28, Pilsen, Czech Republic

Eppinger SD, Browning TR (2012) Design structure matrix methods and applications. MIT Press, Cambridge, MA

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

Evans JH (1959) Basic design concepts. J Am Soc Naval Eng 71(4):671–678

Evbuomwan N, Sivaloganathan S, Jebb A (1996) A survey of design philosophies, models, methods and systems. Proc Inst Mech Eng Part B J Eng Manuf 210(4):301–320

Fernandes JMV (2015) Requirements change in complex product development: Understanding causes, managing uncertainty and planning for change. PhD thesis, Instituto Superior Técnico

Finger S, Dixon JR (1989) A review of research in mechanical engineering design. part I: Descriptive, prescriptive, and computer-based models of design processes. Res Eng Des 1(1):51–67

Ford DN, Sterman JD (1998) Dynamic modeling of product development processes. Syst Dyn Rev 14(1):31–68

Forsberg K, Mooz H, Cotterman H (2005) Visualizing project management: models and frameworks for mastering complex systems, 3rd edn. Wiley, Hoboken, NJ

Freisleben D, Vajna S (2002) Dynamic project navigation: Modelling, improving, and review of engineering processes. In: ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Quebec, Canada, Sep 29–Oct 2, American Society of Mechanical Engineers, vol 2, pp 919–925

French MJ (1999) Conceptual design for engineers, 3rd edn. Springer-Verlag, London

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

Gericke K, Blessing L (2011) Comparisons of design methodologies and process models across disciplines: A literature review. In: Culley SJ, Hicks BJ, McAloone TC, Howard TJ, Clarkson PJ (eds) Proceedings of the 18th International Conference on Engineering Design (ICED 11), Lyngby/Copenhagen, Denmark, Aug 15–19, Design Society, vol 1, pp 393–404

Gericke K, Blessing L (2012) An analysis of design process models across disciplines. In: Marjanović D, Štorga M, Pavković N, Bojčetić N (eds) Proceedings of DESIGN 2012, the 12th International Design Conference, Dubrovnik, Croatia, May 21–24, Design Society, pp 171–180

Gericke K, Eckert CM, Wynn DC (2016) Towards a framework of choices made during the life-cycle of process models. In: Marjanović D, Štorga M, Pavković N, Bojčetić N, Škec S (eds) Proceedings of DESIGN 2016, the 14th International Design Conference, Dubrovnik, Croatia, May 16–19, Design Society, pp 1275–1284

Gero JS, Kannengiesser U (2014) The function-behaviour-structure ontology of design. In: Chakrabarti A, Blessing LTM (eds) An anthology of theories and models of design: philosophy, approaches and empirical explorations. Springer-Verlag, London, pp 263–283

Gero JS (1990) Design prototypes: a knowledge representation schema for design. AI Mag 11(4):26

Gero JS (2000) Computational models of innovative and creative design processes. Technol Forecast Soc Change 64(23):183–196

Gero JS, Kannengiesser U (2004) The situated function-behaviour-structure framework. Des Stud 25(4):373–391

Gonnet S, Henning G, Leone H (2007) A model for capturing and representing the engineering design process. Expert Syst Appl 33(4):881–902

Grabowski H, Lossack RS, Weis C (1996) A design process model based on design working spaces. In: Tomiyama T, Mäntylä M, Finger S (eds) Knowledge intensive CAD, vol 1. Springer US, Boston, pp 244–262

Grabowski H, Lossack RS, El-Mejbri EF (1999) Towards a universal design theory. In: Kals H, van Houten F (eds) Integration of process knowledge into design support systems: proceedings of the 1999 CIRP International Design Seminar, University of Twente, Enschede, The Netherlands, 24–26 March, 1999. Springer, Netherlands, Dordrecht, pp 47–56

Guindon R (1990) Designing the design process: exploiting opportunistic thoughts. Hum Comput Interact 5(2):305–344

Ha A, Porteus E (1995) Optimal timing of reviews in concurrent design for manufacturability. Manag Sci 41(9):1431–1447

Ha S, Suh HW (2008) A timed colored Petri nets modeling for dynamic workflow in product development process. Comput Ind 59(2):193–209

Hales C, Gooch S (2004) Managing engineering design, 2nd edn. Springer-Verlag, London

Hall AD (1962) A methodology for systems engineering. Van Nostrand, New York, NY

Hamraz B, Caldwell NH, Wynn DC, Clarkson PJ (2013) Requirements-based development of an improved engineering change management method. J Eng Des 24(11):765–793

Hatchuel A, Weil B (2003) A new approach of innovative design: an introduction to C-K theory. In: Folkeson A, Gralen K, Norell M, Sellgren U (eds) Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm, Sweden, Aug 19–21, Design Society, pp 109–110

Hatchuel A, Weil B (2009) CK design theory: an advanced formulation. Res Eng Des 19(4):181–192

Hatchuel A, Le Masson P, Weil B (2004) C-K theory in practice: lessons from industrial applications. In: Marjanović D (ed) Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia, May 18–21, Design Society, pp 245–258

Hauser JR, Clausing D (1988) The house of quality. Harvard Business Review May–June 1988

Heisig P, Caldwell NH, Clarkson PJ (2014) Core information categories for engineering design-contrasting empirical studies with a review of integrated models. J Eng Des 25(1–3):88–124

Herroelen W, Leus R (2001) On the merits and pitfalls of critical chain scheduling. J Oper Manag 19(5):559–577

Hillier B, Musgrove J, O'Sullivan P (1972) Knowledge and design. In: Mitchell WJ (ed) Environmental design: research and practice. Proceedings of the EDRA 3/AR 8 Conference, University of California at Los Angeles, January 1972, vol 2, Univ.-Verlag, pp 1–14

Hisarciklilar O, Sheikh OKB, Yadav HA, Thomson V (2013) Improving coordination between aircraft development processes through process mapping and simulation. SAE Int J Aerosp 6(1):87–93

Hoedemaker G, Blackburn J, Wassenhove L (1999) Limits to concurrency. Decis Sci 30(1):1–18

Howard TJ, Culley SJ, Dekoninck E (2008) Describing the creative design process by the integration of engineering design and cognitive psychology literature. Des Stud 29(2):160–180

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

Hubka V (1982) Principles of engineering design. Butterworth Scientific, London, Boston, Sydney, Wellington, Durban, Toronto

Hubka V, Eder WE (1996) Design science: introduction to the needs, scope and organization of engineering design knowledge. Springer-Verlag, London

Hybs I, Gero JS (1992) An evolutionary process model of design. Des Stud 13(3):273–290

ISO/PAS19450 (2015) Automation systems and integration—object-process methodology. International Organization for Standardization

Jin Y, Geslin M (2009) Argumentation-based negotiation for collaborative engineering design. Int J Collab Eng 1(1–2):125–151

Joglekar N, Yassine A, Eppinger SD, Whitney DE (2001) Performance of coupled product development activities with a deadline. Manag Sci 47(12):1605–1620

Jones JC (1963) A method of systematic design. In: Jones J, Thornley D (eds) Conference on design methods: papers presented at the conference on systematic and intuitive methods in engineering, industrial design, architecture and communications, London, England, Sep 1962. Pergamon, Oxford, London, New York and Paris, pp 53–73

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

Karniel A, Reich Y (2011) Managing the dynamics of new product development processes: a new product lifecycle management paradigm. Springer-Verlag, London

Kazakçi A, Hatchuel A, Le Masson P, Weil B (2010) Simulation of design reasoning based on C-K theory: a model and an example application. In: Marjanović D, Štorga M, Pavković N, Bojčetić N (eds) Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia, May 17–20, Design Society, pp 59–68

Kazakçi AO (2009) A formalization of CK design theory based on intuitionist logic. In: Chakrabarti A (ed) ICORD 09: Proceedings of the 2nd International Conference on Research into Design, Bangalore, India, Jan 7–9, Design Society, pp 499–507

Kennedy BM, Sobek DK, Kennedy MN (2014) Reducing rework by applying set-based practices early in the systems engineering process. Syst Eng 17(3):278–296

Kennedy M (2008) Ready, set, dominate: Implement Toyota's set-based learning for developing products and nobody can catch you. Oaklea Press, Richmond, VA

Kerley W, Wynn DC, Eckert C, Clarkson PJ (2011) Redesigning the design process through interactive simulation: a case study of life-cycle engineering in jet engine conceptual design. Int J Serv Oper Manag 10(1):30–51

Klein M (1993) Supporting conflict management in cooperative design teams. Group Decis Negot 2(3):259–278

Kolberg E, Reich Y, Levin I (2014) Designing winning robots by careful design of their development process. Res Eng Des 25(2):157–183

Konda S, Monarch I, Sargent P, Subrahmanian E (1992) Shared memory in design: a unifying theme for research and practice. Res Eng Des 4(1):23–42

Kreimeyer M, Lindemann U (2011) Complexity metrics in engineering design: managing the structure of design processes. Springer-Verlag, Berlin, Heidelberg

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

Kruger C, Cross N (2006) Solution driven versus problem driven design: strategies and outcomes. Des Stud 27(5):527–548

Kusiak A, Wang J (1993a) Decomposition of the design process. J Mech Des 115(4):687–695

Kusiak A, Wang J (1993b) Efficient organizing of design activities. Int J Prod Res 31(4):753–769

Kusiak A, Larson TN, Wang J (1994) Reengineering of design and manufacturing processes. Comput Ind Eng 26(3):521–536

Larman C, Basili VR (2003) Iterative and incremental development: a brief history. Computer 36(6):47–56

Le HN (2013) A transformation-based model integration framework to support iteration management in engineering design. PhD thesis, University of Cambridge

Le Masson P, Dorst K, Subrahamanian E (2013) Special Issue on Design Theory: history, state of the arts and advancements. Res Eng Des 24(2):212–243

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

Lewis K, Mistree F (1998) Collaborative, sequential, and isolated decisions in design. J Mech Des 120(4):643–652

Liker J, Morgan J (2006) The Toyota way in services: the case of lean product development. Acad Manag Perspect 20(2):5–20

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

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

Loch C, Mihm J, Huchzermeier A (2003) Concurrent engineering and design oscillations in complex engineering projects. Concurr Eng Res Appl 11(3):187–199

Loch CH, Terwiesch C, Thomke S (2001) Parallel and sequential testing of design alternatives. Manag Sci 47(5):663–678

Lyneis JM, Ford DN (2007) System dynamics applied to project management: a survey, assessment, and directions for future research. Syst Dyn Rev 23(2/3):157–189

Maffin D, Alderman N, Braiden P, Hills B, Thwaites A (1995) Company classification: a new perspective on modelling the engineering design and product development process. J Eng Des 6(4):275–289

Maher ML (2000) A model of co-evolutionary design. Eng Comput 16(3):195–208

Maher ML, Poon J (1996) Modeling design exploration as coevolution. Microcomput Civil Eng 11(3):195–209

Maier AM, Wynn DC, Howard TJ, Andreasen MM (2014) Perceiving design as modelling: a cybernetic systems perspective. In: Chakrabarti A, Blessing LTM (eds) An anthology of theories and models of design: philosophy, approaches and empirical explorations. Springer-Verlag, London, pp 133–149

Maier JF, Eckert CM, Clarkson PJ (2017) Model granularity in engineering design—concepts and framework. Des Sci 3(1):1–29

Malone TW, Crowston K (1994) The interdisciplinary study of coordination. ACM Comput Surv 26(1):87–119

March L (1976) The logic of design and the question of value. In: March L (ed) The architecture of form. Cambridge University Press, Cambridge

Marples DL (1961) The decisions of engineering design. IRE Trans Eng Manag 2:55–71

Maurer M (2017) Complexity management in engineering design—a primer. Springer Vieweg, Berlin, Heidelberg

Mayer RJ, Menzel CP, Painter MK, Dewitte PS, Blinn T, Perakath B (1995) Information integration for concurrent engineering (IICE) IDEF3 process description capture method report, KBSI-IICE-90-STR-01-0592-02. Tech. rep, Knowledge Based Systems, Incorporated, College Station, Texas, USA

McMahon CA, Xianyi M (1996) A network approach to parametric design integration. Res Eng Des 8(1):14–31

McManus HL (2005) Product development value stream mapping (PDVSM) manual. Lean Aerospace Initiative, Massachusetts Institute of Technology, Cambridge, MA

Melão N, Pidd M (2000) A conceptual framework for understanding business processes and business process modelling. Inf Syst J 10(2):105–129

Mihm J, Loch C, Huchzermeier A (2003) Problem-solving oscillations in complex engineering projects. Manag Sci 46(6):733–750

Moen RD, Norman CL (2010) Circling back. Qual Prog 43(11):22–28

Mohd Saad N, Al-Ashaab A, Maksimovic M, Zhu L, Shehab E, Ewers P, Kassam A (2013) A3 thinking approach to support knowledge-driven design. Int J Adv Manuf Technol 68(5):1371–1386

Muller D, Reichert M, Herbst J, Poppa F (2007) Data-driven design of engineering processes with COREPROModeler. In: Reddy S (ed) Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2007), Paris, France, Jun 18–20, IEEE Computer Society, pp 376–378

Narahari Y, Viswanadham N, Kumar VK (1999) Lead time modeling and acceleration of product design and development. IEEE Trans Robot Autom 15(5):882–896

Nelson RG, Azaron A, Aref S (2016) The use of a GERT based method to model concurrent product development processes. Eur J Oper Res 250(2):566–578

O'Donovan BD, Eckert CM, Clarkson PJ (2004) Simulating design processes to assist design process planning. In: ASME 2004

International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Salt Lake City, Utah, Sep 28–Oct 2, American Society of Mechanical Engineers, vol 3a, pp 503–512
O'Donovan BD, Eckert CM, Clarkson PJ, Browning TR (2005) Design planning and modelling. In: Clarkson PJ, Eckert CM (eds) Design process improvement: a review of current practice. Springer-Verlag, London, pp 60–87
Oppenheim B (2004) Lean product development flow. Syst Eng 7(4):352–376
Ottosson S (2004) Dynamic product development—DPD. Technovation 24(3):207–217
Ouertani MZ, Gzara L (2008) Tracking product specification dependencies in collaborative design for conflict management. Comput Aided Des 40(7):828–837
Pahl G, Beitz W, Feldhusen J, Grote K-H (2007) Engineering design: a systematic approach, 3rd edn. Springer-Verlag, London
Park H, Cutkosky MR (1999) Framework for modeling dependencies in collaborative engineering processes. Res Eng Des 11(2):84–102
Parry G, Turner C (2006) Application of lean visual process management tools. Prod Plan Control 17(1):77–86
Pich MT, Loch CH, De Meyer A (2002) On uncertainty, ambiguity, and complexity in project management. Manag Sci 48(8):1008–1023
Pidd M (1999) Just modeling through: a rough guide to modeling. Interfaces 29(2):118–132
Pocock J (1962) PERT as an analytical aid for program planning—its payoff and problems. Oper Res 10(6):893–903
Prasad B (1996a) Concurrent engineering fundamentals, vol 1. Prentice Hall, Englewood Cliffs
Prasad B (1996b) Toward definitions of a concurrent product design, development, and delivery (PD^3) system. Concurr Eng 4(2):102–109
Pritsker A (1966) GERT: Graphical evaluation and review technique. Memorandum RM-4973-NASA April 1966
Pugh S (1991) Total design: integrated methods for successful product engineering. Addison-Wesley, Boston, MA
Rangan RM, Rohde SM, Peak R, Chadha B, Bliznakov P (2005) Streamlining product lifecycle processes: a survey of product lifecycle management implementations, directions, and challenges. J Comput Inf Sci Eng 5(3):227–237
Raudberget D (2010) Practical applications of set-based concurrent engineering in industry. Strojniski Vestnik/J Mech Eng 56(11):685–695
Rawson KJ, Tupper EC (2001) Basic ship theory, combined volume. 5th ed. Butterworth-Heinemann, Oxford
Reich Y (1995) A critical review of general design theory. Res Eng Des 7(1):1–18
Reichelt K, Lyneis J (1999) The dynamics of project performance: benchmarking the drivers of cost and schedule overrun. Eur Manag J 17(2):135–150
Roemer TA, Ahmadi R (2004) Concurrent crashing and overlapping in product development. Oper Res 52(4):606–622
Romero F, Company P, Agost MJ, Vila C (2008) Activity modelling in a collaborative ceramic tile design chain: an enhanced IDEF0 approach. Res Eng Des 19(1):1–20
Roozenburg N, Cross N (1991) Models of the design process: integrating across the disciplines. Des Stud 12(4):215–220
Roozenburg NF, Eekels J (1995) Product design: fundamentals and methods. Wiley, Chichester
Rother M, Shook J (2003) Learning to see: value stream mapping to add value and eliminate muda. Lean Enterprise Institute, Cambridge, MA
Rouibah K, Caskey K (2003) A workflow system for the management of inter-company collaborative engineering processes. J Eng Des 14(3):273–293
Saaty R (1987) The analytic hierarchy process—what it is and how it is used. Math Model 9(3):161–176
Salustri FA (2014) Reformulating CK theory with an action logic. In: Gero JS (ed) Design computing and cognition '12. Springer, Dordrecht, pp 433–450
Schlick CM, Duckwitz S, Schneider S (2013) Project dynamics and emergent complexity. Comput Math Org Theory 19(4):480–495
Shah JJ, Jeon DK, Urban SD, Bliznakov P, Rogers M (1996) Database infrastructure for supporting engineering design histories. Comput Aided Des 28(5):347–360
Shah JJ, Vargas-Hernandez NOE, Summers JD, Kulkarni S (2001) Collaborative sketching (C-Sketch)—an idea generation technique for engineering design. J Creat Behav 35(3):168–198
Shapiro D, Curren MD, Clarkson PJ (2016) DPCM: a method for modelling and analysing design process changes based on the applied signposting model. J Eng Des 27(11):785–816
Sharafi A, Wolfenstetter T, Wolf P, Krcmar H (2010) Comparing product development models to identify process coverage and current gaps: A literature review. In: Proceedings of IEEM2010, The IEEE International Conference on Industrial Engineering and Engineering Management, Macao, China, Dec 7–10, IEEE, pp 1732–1736
Sharon A, Dori D (2015) A project-product model-based approach to planning work breakdown structures of complex system projects. IEEE Syst J 9(2):366–376
Sharon A, de Weck OL, Dori D (2013) Improving project-product lifecycle management with model-based design structure matrix: A joint project management and systems engineering approach. Syst Eng 16(4):413–426
Sim SK, Duffy AHB (2003) Towards an ontology of generic engineering design activities. Res Eng Des 14(4):200–223
Siyam GI, Wynn DC, Clarkson PJ (2015) Review of value and lean in complex product development. Syst Eng 18(2):192–207
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
Smith RP, Tjandra P (1998) Experimental observation of iteration in engineering design. Res Eng Des 10(2):107–117
Smithers T (1998) Towards a knowledge level theory of design process. In: Gero J, Sudweeks F (eds) Artif Intell Des '98. Springer, Netherlands, pp 3–21
Sobek DK, Ward A, Liker J (1999) Toyota's principles of set-based concurrent engineering. Sloan Manag Rev 40(2):67–83
Sosa ME, Eppinger SD, Rowles CM (2004) The misalignment of product architecture and organizational structure in complex product development. Manag Sci 50(12):1674–1689
Sosa R, Gero JS (2016) Multi-dimensional creativity: a computational perspective. Int J Des Creat Innov 4(1):26–50
Srinivasan V, Chakrabarti A (2010) An integrated model of designing. J Comput Inf Sci Eng 10(3):031,013
Steward DV (1981) The design structure system: A method for managing the design of complex systems. IEEE Trans Eng Manag EM-28(3):71–74
Suh NP (1990) The principles of design. Oxford University Press, New York
Suss S, Thomson V (2012) Optimal design processes under uncertainty and reciprocal dependency. J Eng Des 23(10–11):826–848
Takeda H, Veerkamp P, Yoshikawa H (1990) Modeling design processes. AI Mag 11(4):37
Taura T, Kubota A (1999) A study on engineering history base. Res Eng Des 11(1):45–54
Taylor BW, Moore LJ (1980) R&D project planning with Q-GERT network modeling and simulation. Manag Sci 26(1):44–59

Tomiyama T (1994) From general design theory to knowledge-intensive engineering. Artif Intell Eng Des Anal Manuf 8(04):319–333

Tomiyama T, Kiriyama T, Takeda H, Xue D, Yoshikawa H (1989) Metamodel: a key to intelligent CAD systems. Res Eng Des 1(1):19–34

Tomiyama T, Gu P, Jin Y, Lutters D, Kind C, Kimura F (2009) Design methodologies: industrial and educational applications. CIRP Ann Manuf Technol 58(2):543–565

Turner R (2007) Toward agile systems engineering processes. Crosstalk J Defense Softw Eng 2007:11–15

Ullman D (2015) The mechanical design process, 5th edn. McGraw-Hill Education, New York, NY

Ullman DG, Dietterich TG, Stauffer LA (1988) A model of the mechanical design process based on empirical data. Artif Intell Eng Des Anal Manuf 2(1):33–52

Ulrich KT, Eppinger SD (2015) Product design and development, 6th edn. McGraw-Hill Education, New York, NY

USAF (1981) ICAM Architecture Part II—Volume IV—Function Modeling Manual (IDEF0), AFWAL-TR-81-4023. Tech. rep., Materials Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, USA

Vajna S (2005) Workflow for design. In: Clarkson PJ, Eckert CM (eds) Design process improvement: a review of current practice. Springer-Verlag, London, pp 366–385

Vajna S, Burchardt C (1998) Dynamic development structures of integrated product development. J Eng Des 9(1):3–15

Vajna S, Clement S, Jordan A, Bercsey T (2005) The autogenetic design theory: an evolutionary view of the design process. J Eng Des 16(4):423–440

Van der Aalst WM (1998) The application of Petri nets to workflow management. J Circ Syst Comput 8(1):21–66

VDI2206 (2004) Design methodology for mechatronic systems (VDI2206). Verein Deutscher Ingenieure

VDI2221 (1987) Systematic approach to the design of technical systems and products (VDI2221). Verein Deutscher Ingenieure

Weber C (2014) Modelling products and product development based on characteristics and properties. In: Chakrabarti A, Blessing LTM (eds) An anthology of theories and models of design: philosophy, approaches and empirical explorations. Springer-Verlag, London, pp 327–352

Weber C, Werner H, Deubel T (2003) A different view on product data management/product life-cycle management and its future potentials. J Eng Des 14(4):447–464

Whitney DE (2004) Mechanical assemblies: their design, manufacture, and role in product development. Oxford University Press Inc, New York, NY

Wynn DC (2007) Model-based approaches for process improvement in complex product development. PhD thesis, University of Cambridge

Wynn DC, Clarkson PJ (2005) Models of designing. In: Clarkson PJ, Eckert CM (eds) Design process improvement: a review of current practice. Springer-Verlag, London, pp 34–59

Wynn DC, Eckert CM (2017) Perspectives on iteration in design and development. Res Eng Des 28(2):153–184

Wynn DC, Eckert CM, Clarkson PJ (2006) Applied Signposting: a modeling framework to support design process improvement. In: ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Philadelphia, Pennsylvania, USA, Sep 10–13, American Society of Mechanical Engineers, vol 4a, pp 553–562

Wynn DC, Maier AM, Clarkson PJ (2010) How can PD process modelling be made more useful? an exploration of factors which influence modelling utility. In: Marjanović D, Štorga M, Pavković N, Bojčetić N (eds) Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia, May 17–20, Design Society, pp 511–522

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

Wynn DC, Caldwell NHM, Clarkson PJ (2014) Predicting change propagation in complex design workflows. J Mech Des 136(8):081009, 13

Xin Chen HL, Moullec ML, Ball N, Clarkson PJ (2016) Improving design resource management using Bayesian network embedded in task network method. In: ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Charlotte, North Carolina, USA, Aug 21–24, American Society of Mechanical Engineers, vol 7, p V007T06A034, p 15

Yassine A (2004) An introduction to modeling and analyzing complex product development processes using the Design Structure Matrix (DSM) method. Urbana 51(9):1–17

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

Yassine A, 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

Yoshikawa H (1981) General design theory and a CAD system. In: Sata T, Warman E (eds) Man-Machine communication in CAD/CAM: Proceedings of the IFIP WG5.2-5.3 Working Conference held in Tokyo, Japan, 2–4 October 1980, North-Holland Publishing Company

Zachman J (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292

Zeng Y (2002) Axiomatic theory of design modeling. J Integr Des Process Sci 6(3):1–28

Zeng Y, Cheng G (1991) On the logic of design. Des Stud 12(3):137–141

Zeng Y, Yao S (2009) Understanding design activities through computer simulation. Adv Eng Inf 23(3):294–308

Zhang X, Hao Y, Thomson V (2015) Taking ideas from paper to practice: a case study of improving design processes through detailed modeling and systematic analysis. IFAC-PapersOnLine 48(3):1043–1048