Is Designing Independent of Domain? Comparing Models of Engineering, Software and Service Design

UDO KANNENGIESSER
Metasonic GmbH, Germany

and

JOHN S GERO
University of North Carolina at Charlotte, USA

Abstract. This paper presents a quantitative method for analysing process models of designing independently of the specific design domain. The method uses the situated Function-Behaviour-Structure (FBS) framework as the basis for a simulation model of a designer acting according to these models. The results of these simulations are sequences of design issues that are analysed using cumulative occurrence graphs with associated quantitative measures. The paper illustrates the approach by analysing and comparing three models of designing from different domains: Pahl and Beitz' model of engineering design, the Rational Unified Process of software design, and a model of Design for Six Sigma in service design. The quantitative results indicate some commonalities across the different models. These commonalities are related to the start of cognitive effort spent on design issues, the continuity of the cognitive effort throughout the design process, and the constancy of the speed with which design issues are generated.

1. Introduction

Designing is a complex activity that has attracted a significant amount of attention from different research domains, trying to demystify its manifold processes. One of the biggest challenges is to define designing as a unique activity while it is used in a vast range of domains such as engineering, software, graphical interfaces, and electronics, to name a few. Understanding the commonalities amongst different expressions of designing is a foundational step in developing a universal understanding of design (Asimow 1962; Lawson 1980; Cross 1982; Dym 1994; Visser 2009).

We hypothesise that designing is an act that is independent of the domain of its application, in the sense that different domains have the same understanding of this act even if they use different terms to describe it. This paper presents an approach to testing this hypothesis based on analys-

ing and comparing models of designing from different domains. There have been similar efforts in the past. For example, in 1998 an international workshop organized by Grabowski et al. (1998) brought together design theorists from different disciplines, aiming to build a unified or universal design theory. Discussions concentrated on finding out whether differences between the models were caused by different concepts or just by different terms for the same concept. Such discussions have continued until today (Sim and Duffy 2003; Frey and Dym 2006; Boon and Knuuttila 2009; Vermaas 2009; Eder 2012; Chakrabarti and Blessing 2014; Vermaas 2014; Lindemann 2014).

Approaches to extracting commonalities across different models of designing have been limited to qualitative analyses. In this paper we propose a quantitative approach to analysing and comparing models of designing. It is based on simulations of the design process using the domain-independent function-behaviour-structure (FBS) ontology and its derivative, the situated FBS framework. The simulation models are constructed by mapping the models of designing onto the 20 processes defined in the situated FBS framework and aggregating them to the six design issues of: requirement issues, function issues, expected behaviour issues, structure behaviour issues, structure issues, and description issues. The cumulative occurrence of the six design issues over the course of a simulation is analysed in terms of their first occurrence (representing the start of cognitive effort spent on design issues), their continuity (characterising the cognitive effort spent on design issues towards the end of designing), their linearity (representing the constancy of the speed with which design issues are produced) and their slope (representing the rate at which design issues are produced). The quantitative approach presented in this paper is applied to Pahl and Beitz' model of engineering design, the Rational Unified Process of software design, and the Design for Six Sigma model of service design.

This paper is structured as follows: Section 2 presents the three models of designing that will be used to demonstrate the approach and outlines their common overall process structure qualitatively. Section 3 develops a simulation model for these domain-specific models of designing based on the steps contained within them. Section 4 presents the results derived from running the simulation for each of the three models of designing. Section 5 describes the commonalities found, and Section 6 discusses some conclusions that can be drawn from the results. Appendices include the situated FBS framework and the mappings between the three models and the FBS design issues.

2. Three Domain-Specific Models of Designing

Domain-specific models differ from each other mostly in the concepts they use for describing the respective artefacts to be designed. These models commonly represent designing as a phase-based activity (Tate and Nordlund 1996) where the state of the design gradually progresses from abstract to concrete. We chose three phase-based models of designing from disparate design domains as a basis for our analyses: engineering design, software design, and service design.

Engineering design is a design discipline with a long tradition in developing models of designing. One of the most detailed and established models in this discipline is Pahl and Beitz' (2007) *Systematic Approach*, which was first published in its German edition in 1977. It describes designing as a sequence of four phases: (1) Task Clarification, (2) Conceptual Design, (3) Embodiment Design, and (4) Detail Design. Task Clarification is concerned with collecting, formulating and documenting the requirements of the product to be designed. Conceptual Design aims to identify the basic principles and outline of a design solution (or concept). Embodiment Design then elaborates the design into a layout that satisfies various technical and economic criteria. Detail Design finalises the design and prepares production documents. Each of the four phases comprises a sequence of activities that may be executed iteratively. After every phase a "decision-making step" is performed to assess the results of the phase and decide whether the subsequent phase can be started or whether the phase needs to reiterate. Here, "[t]he smallest possible iteration loop is desirable" (*ibid*, p. 129). Pahl and Beitz do not explicitly exclude iterations across different phases. On the other hand, the "phase-based" character of the Systematic Approach clearly favours a "waterfall" view where iterations are to occur only within a phase (Tate and Nordlund 1996; Unger and Eppinger 2011). Table 1 shows the phases of Pahl and Beitz' Systematic Approach, and the activities associated with each phase.

Table 1 Pahl and Beitz' (2007) Systematic Approach

Phases	Activities
1. Task Clarification	1.1 Define basic market demands 1.2 Define attractiveness demands of the market segment 1.3 Document customer-specific technical performance requirements 1.4 Refine and extend the requirements using the checklist and scenario planning 1.5 Determine demands and wishes
2. Conceptual	2.1 Abstract to identify the essential problems

Design	2.2 Establish function structures: overall function – subfunctions 2.3 Search for working principles that fulfil the subfunctions 2.4 Combine working principles into working structures 2.5 Select suitable combinations 2.6 Firm up into principle solution variants 2.7 Evaluate variants against technical and economic criteria
3. Embodiment Design	3.1 Identify embodiment-determining requirements 3.2 Produce scale drawings of spatial constraints 3.3 Identify embodiment-determining main function carriers 3.4 Develop preliminary layouts and form designs for the embodiment-determining main function carriers 3.5 Select suitable preliminary layouts 3.6 Develop preliminary layouts and form designs for the remaining main function carriers 3.7 Search for solutions to auxiliary functions 3.8 Develop detailed layouts and form designs for the main function carriers ensuring compatibility with the auxiliary function carriers 3.9 Develop detailed layouts and form designs for the auxiliary function carriers and complete the overall layouts 3.10 Evaluate against technical and economic criteria 3.11 Optimise and complete form designs 3.12 Check for errors and disturbing factors 3.13 Prepare preliminary parts lists and production documents
4. Detail Design	4.1 Finalise details; complete detail drawings 4.2 Integrate into overall layout drawings, assembly drawings and parts lists 4.3 Complete production documents with production, assembly, transport and operating instructions 4.4 Check all documents for standards, completeness and correctness

The discipline of software design has also brought about several models of designing. Here, one of the most widely used models is the *Rational Unified Process* (RUP). Although it was primarily developed as a commercial product, its basic concepts outlined by Kruchten (2004) form a publicly available and highly cited model of designing. RUP defines the following phases for software design processes: (1) Inception, (2) Elaboration, (3) Construction, and (4) Transition. Inception deals with understanding the requirements and defining the scope of the design. Elaboration specifies and prototypes the main features and architecture of the software design solution. Construction elaborates this solution by developing the

complete set of features and implementing all the components of the software. Transition focuses on verifying design quality, manufacturing, and delivering the software to the user. Kruchten (2004) suggests this four-phase process be executed iteratively. He also suggests that the specific activities within each phase are to be configured depending on the needs of the individual design project. On the other hand, he describes "typical iteration plans" (*ibid*, Chapter 16) that can be viewed as a representative sequence of activities that is likely to cover most instances of software design processes. Table 2 summarises the phases and activities in such a "typical" configuration of RUP.[1]

Table 2 Kruchten's (2004) Rational Unified Process

Phases	Activities
1. Inception	1.1 Analyze the problem 1.2 Understand stakeholder needs 1.3 Define the system 1.4 Manage the scope of the system 1.5 Refine the system definition
2. Elaboration	2.1 Decide which use cases and scenarios will drive the development of the architecture 2.2 Understand this driver in detail and inspect the results 2.3 Reconsider use cases and risks 2.4 Prototype the user interface 2.5 Find obvious classes, do initial subsystem partitioning, and look at use cases in detail 2.6 Refine and homogenize classes and identify architecturally significant ones; inspect results 2.7 Consider the low-level package partitioning 2.8 Adjust to the implementation environment, decide the design of the key scenarios, and define formal class interfaces; inspect results 2.9 Consider concurrency and distribution of the architecture 2.10 Inspect the architectural design 2.11 Consider the physical packaging of the architecture 2.12 Plan the integration 2.13 Plan integration tests and system tests 2.14 Implement the classes and integrate 2.15 Integrate the implemented parts 2.16 Assess the executable architecture

[1] For the Inception phase we use the workflow defined for the requirements discipline and omit the design project management activities that are included in Kruchten's "typical" Inception phase. We view these management activities as beyond the scope of a model of designing. For the Transition phase, where there are no "typical" activities defined, we use Kruchten's deployment workflow.

3. Construction	3.1 Plan system-level integration 3.2 Plan and design system-level test 3.3 Refine use-case realizations 3.4 Plan and design integration tests at the subsystem and system levels 3.5 Develop code and test unit 3.6 Plan and implement unit test 3.7 Test unit within a subsystem 3.8 Integrate a subsystem 3.9 Test a subsystem 3.10 Release a subsystem 3.11 Integrate the system 3.12 Test integration 3.13 Test the system
4. Transition	4.1 Plan deployment 4.2 Develop support material 4.3 Produce deployment unit 4.4 Beta test product

Service design is a more recent discipline with few existing process models. One of them is *Design for Six Sigma* (DFSS), which has been used to describe both designing products and designing services (or processes). One of the many variants of DFSS that is specific to designing services is the ICOV (Identify-Conceptualize-Optimize-Validate) model presented by El-Haik and Roy (2005). We will refer to this model as DFSS-ICOV in this paper. It proposes the following phases: (1) Identify, (2) Conceptualize, (3) Optimize, and (4) Validate. The Identify phase collects and analyses the requirements for the service to be designed, by listening to both the "voice of the customer" and the "voice of the business". The Conceptualize phase determines the technical requirements and basic components of the service. The Optimize phase aims to configure the service in a way to achieve the best possible performance. The Validate phase tests and refines the service and prepares its launch. At the end of every phase in DFSS-ICOV there is a review to decide whether to proceed to the next phase or whether to rework some decisions. Table 3 shows the phases and activities described in this model.

Table 3 El-Haik and Roy's (2005) Identify-Conceptualize-Optimize-Validate model (Design for Six Sigma)

Phases	Activities
1. Identify	1.1 Idea creation 1.2 Voice of the customer and business
2. Conceptualize	2.1 Concept development

	2.2 Preliminary design
3. Optimize	3.1 Design optimization
4. Validate	4.1 Verification 4.2 Launch readiness

While there are obvious domain-specific differences between the three models, we can already extract a first commonality: All three models use four sequential phases with similar goals, Table 4. As designing proceeds through the four phases, its focus ultimately shifts from the design problem (phase 1) to the design solution (phase 4), with two intermediate stages: One stage (phase 2) generates a list of general concepts that have the potential of being used as starting points for synthesis of variations ("concept structure"). The other stage (phase 3) turns these general concepts into specific solutions with respect to formulated goals, constraints or resources ("solution structure"). This general four-phase model is consistent with the widely held understanding of designing as a progression from the abstract to the concrete (Roozenburg and Cross 1991; Welch and Dixon 1994; Hubka and Eder 1996).

Table 4 Common goals of the individual phases in Pahl and Beitz' Systematic Approach, Kruchten's RUP, and El-Haik and Roy's DFSS-ICOV

Phase	Systematic Approach	RUP	DFSS-ICOV	Overall goal
1	Task Clarification	Inception	Identify	Understanding & defining the design problem
2	Conceptual Design	Elaboration	Conceptualize	Generating a concept structure
3	Embodiment Design	Construction	Optimize	Generating a solution structure
4	Detail Design	Transition	Validate	Finalising & delivering the design solution

Despite these commonalities there is also a significant difference between the three models: While the number of activities in the Systematic Approach (29) and RUP (35) is quite similar, the number of activities in DFSS-ICOV (7) is much smaller. This raises doubts about the usefulness of choosing DFSS-ICOV for comparison against the other models of designing. In Section 3.2 we show that DFSS-ICOV will become more fine-grained after applying the FBS coding scheme to become more compara-

ble against the other models of designing. In addition, Section 3.3 will show that the measures we use for analysing and comparing different models are independent of the number of activities or steps described within a model.

3. Developing a Simulation Model

Models of designing are generally understood as guidelines to be used by designers when tackling a design task. If we can describe the activities of a designer who follows the guidelines provided by a specific model, we can simulate the design process represented in the model. This Section presents how such a simulation model can be produced in two steps: generalising the concepts and terms used by a specific model of designing into FBS design issues, and mapping the model onto the situated FBS framework.

3.1 Generalising Model-Specific Concepts into FBS Design Issues

Each of the three models of designing describes sequences of activities within the four design phases. The models differ not only in the number of these activities, but also in the terms and concepts they use to describe the output of every activity. For a more detailed analysis, we need to map the specific concepts used in the models onto a uniform, generic coding schema. One such schema is the FBS design issue schema that has previously been used for analysing design protocols (Gero and McNeill 1998; Kan and Gero 2005). It consists of six design issues: requirements, function, expected behaviour, behaviour derived from structure (or, shorthand, structure behaviour), structure, and description.

Requirements: includes all expressions of customer or market needs, demands, wishes and constraints that are explicitly provided to the designers at the outset of a design task. For example, requirement issues include "technical performance requirements […] articulated by the customer" (Pahl and Beitz 2007, p. 150), "stakeholder requests" (Kruchten 2004, p. 166), and "customer needs and wants" (El-Haik and Roy 2005, p. 84).

Function: includes teleological representations that can cover any expression related to potential purposes of the artefact. These representations may be flow-based or state-based (Chittaro and Kumar 1998). Unlike requirement issues, function issues are not directly provided to the designer; they are generated by the designer based on interpretations of requirement issues. Function issues in the Systematic Approach include "the intended input/output relationship of a system" (Pahl and Beitz 2007, p. 31) and some examples of needs related to safety, aesthetics or economic proper-

ties. Function issues in RUP include the notion of a use case as a "sequence of actions a system performs that yields an observable result of value to a particular actor" (Kruchten 2004, p. 98), and some "nonfunctional requirements" that "deliver the desired quality to the end user" (*ibid*, p. 159). Function issues in DFSS-ICOV include "service and process functional requirements" that are derived from those requirements provided by the customer (El-Haik and Roy 2005, p. 87).

Expected Behaviour: includes attributes that describe the artefact's expected interaction with the environment. They can be used as guidance and measurable assessment criteria for potential design solutions. Expected behaviour issues in the Systematic Approach include "physical effects" describing the "working principles" of the interactions between different parts of the design object (Pahl and Beitz 2007, p. 40), as well as "technical, economic and safety criteria" used for design evaluation (*ibid*, p. 193). Similarly, Expected behaviour issues in RUP are captured by the "design model" that "consists of a set of collaborations of model elements that provide the behaviour of the system" (Kruchten 2004, p. 177), and "measurable testing goals" (*ibid*, p. 253) that are often subsumed in "nonfunctional requirements". Expected behaviour issues in DFSS-ICOV include "CTSs (critical-to-satisfaction requirements, also known as big *Y*s)" (El-Haik and Roy 2005, p. 33) and some "functional requirements" such as the (expected) "service time" (*ibid*, p. 96). CTSs are considered expected behaviour (rather than function) because they specify measurable objectives with "acceptable performance levels" (Yang and El-Haik 2003).

Structure Behaviour (or "Behaviour derived from Structure"): includes those attributes of the artefact that are measured, calculated or derived from observation of a specific design solution and its interaction with the environment. Instances of structure behaviour must be of the same type as instances of expected behaviour, so as to allow for the comparison and evaluation of design solutions. As a result, structure behaviour issues cover the same notions in the three models of designing as outlined for expected behaviour issues.

Structure: includes the components of an artefact and their relationships. They can appear either as a "concept structure" or a "solution structure", which are the outputs of phases 2 and 3 in Table 1. The former includes Pahl and Beitz' (2007, p. 40) "working surfaces" and "working materials", Kruchten's (2004, p. 174) "classes and subsystems", and El-Haik and Roy's (2005, p. 6) "design parameters". The latter includes Pahl and Beitz' (2007, p. 227) "layout" and "form", Kruchten's (2004, p. 256) "code", and El-Haik and Roy's (2005, p. 7) "detail designs".

Description: includes any form of design-related representations pro-

duced by a designer, at any stage of the design process. The descriptions presented in the Systematic Approach include sketches, CAD models, requirements lists, physical prototypes, calculations, and other documentation produced by mechanical engineers. Descriptions in RUP include storyboards, UML models, code files, test plans and other representations produced by software designers. Descriptions in DFSS-ICOV include House of Quality diagrams, FMEA worksheets, process maps, and concept selection matrices, among many others.

3.2 Mapping the Models of Designing onto the Situated FBS Framework

Every activity described in the three models of designing is concerned with generating one or more design issues. These activities may be mapped onto the eight fundamental processes defined in the FBS framework (Gero 1990), labelled 1 to 8 in Fig. 1.:

1. *Formulation*: transforms requirements into functions (R → F), and functions into expected behaviour (F → Be).
2. *Synthesis*: transforms expected behaviour into structure (Be → S).
3. *Analysis*: transforms structure into structure behaviour (S → Bs).
4. *Evaluation*: compares expected behaviour with structure behaviour (Be ↔ Bs).
5. *Documentation*: transforms structure into a description (S → D).
6. *Reformulation type 1*: transforms structure into new structure (S → S').
7. *Reformulation type 2*: transforms structure into new expected behaviour (S → Be').
8. *Reformulation type 3*: transforms structure into new function (S → F' via Be).

For simulating the design process, however, these processes are still too coarse-grained as they do not include the situation in which they are performed. A more detailed view is provided by the situated FBS (sFBS) framework (see Appendix A) that represents designing as the interaction of a designer with the design situation (Gero and Kannengiesser 2004). This framework defines 20 discrete processes that include a number of cognitive and physical activities, such as the interpretation of requirement lists and design representations, the reflection on current or past design experiences, the decision-making regarding the current design state space, and physical actions including sketching, calculating and documenting.

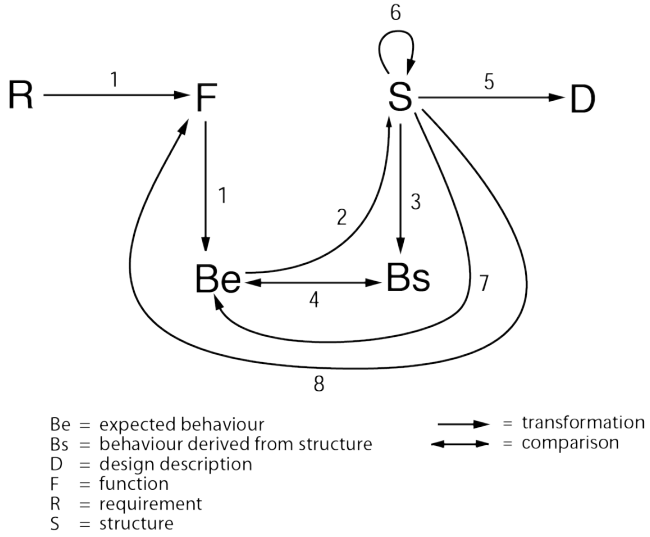

Fig. 1 The FBS framework

Mapping the activities described in a model of designing onto the sFBS framework allows considering the designer's situated interactions in the simulation model. At the same time, the basic representation of designing in terms of the six design issues is maintained. This is because the results of executing the 20 processes are specialised classes of design issues that can be aggregated back to the original six categories. The aggregation of the 20 sFBS processes to the six FBS design issues is shown in Table 5.

Table 5 The results of each of the 20 sFBS processes (labelled based on the sFBS framework shown in Appendix A) are aggregated to the six FBS design issues

sFBS process	FBS design issue
1	R
2	R
3	R
4	F
5	Be or Bs (*)
6	S
7	F
8	Be
9	S
10	Be
11	S
12	D

13	S
14	Bs
15	--- (**)
16	F
17	D
18	D
19	Be or Bs (*)
20	F

* depending on whether the behaviour produced in these processes is interpreted as expected/desired or "actual"/emerging
** This process produces no design issue

The mappings onto the sFBS framework require some interpretation of each model of designing in terms of elementary steps and the logical sequences of these steps. The three models presented in Section 2 provide sufficient elaboration and illustration to support this interpretation for most of their defined activities. Take the first activity, "Define basic market demands", described within Pahl and Beitz' design phase of Task Clarification. This activity requires as input the interpretation of a "development order" or "product proposal" that contains the product's desired "functionality and performance", which in the FBS design issue system is a requirement issue (interpreted by process 1 in the sFBS framework). Next, "basic market demands", such as "suitable for tropical conditions" and "P > 20 kW" (Pahl and Beitz 2007, p. 147), are constructed by the designer as "implicit requirements, i.e. they are not articulated by the customer" (*ibid*, p. 150). We map these market demands onto function and expected behaviour issues (constructed by processes 4 and 5). They are compiled in a "requirements list" and "Quality Function Deployment (QFD)" diagrams (*ibid*, p. 145) that represent description issues (produced by processes 18 and 17). As shown in Table 6, these mappings result in five elementary design steps, each of which produces one design issue, and their logical sequence. (More detailed comments for each of the mappings in the three models of designing can be found in Appendices B, C and D.)

Table 6 The steps involved in Pahl and Beitz' activity of "Define basic market demands" and their mappings onto the FBS design issue system and the sFBS framework

Design step	Pahl and Beitz' description	Process in sFBS (label)	FBS design issue
1	Receive "development order" or "product proposal"	Interpret functional requirements (1)	Requirement

2	Identify basic market demands	Construct functions not explicitly stated (4)	Function
3		Construct expected behaviours not explicitly stated (5)	Expected Behaviour
4	Produce QFD diagrams and requirements list	Produce external representations of function (18)	Description
5		Produce external representations of expected behaviour (17)	

This method of coding and mapping was applied to all three models of designing, which was done in consensus between the two authors of this paper who are experts in FBS coding. The Systematic Approach has 87 mappings, RUP has 100 mappings, and DFSS-ICOV has 41 mappings.

The three sets of mappings of elementary steps can be viewed as a basis for simulation models that need to be complemented with assumptions regarding:

1. the number of occurrences of every elementary step, and
2. the number of iterations within a design phase (we assume that no cross-phase iterations will occur, given the "waterfall" nature of the models)

The first of these assumptions cannot be made without knowledge of specific instances of designing including knowledge about the novelty and complexity of the design task. Staying on the model level rather than the instance level, our working assumption is that every elementary step occurs only once within the same iteration. This assumption is used for each of the three models, and will be revisited in the discussion of results.

The second assumption is similarly based on task- and designer-specific knowledge that is not available at this general level. However, in the case of RUP, Kruchten (2004, p. 133) states that there are three typical scenarios regarding the number of iterations for each of the four phases within RUP (phase 1: inception; phase 2: elaboration; phase 3: construction; phase 4: transition; see Table 4). These scenarios are shown in Table 7. They account for various influences on the design process, including task-related, technical, organisational, personal, market-related and other factors.

Table 7 The number of phase iterations in three typical scenarios of RUP (Kruchten 2004, p. 133)

	Scenario 1 (S1)	Scenario 2 (S2)	Scenario 3 (S3)
Phase 1	0	1	1
Phase 2	1	2	3
Phase 3	1	2	3
Phase 4	1	1	2

For the Systematic Approach and DFSS-ICOV no concrete scenarios are detailed in the literature. Based on the high-level structural similarity of our three models (as shown in Section 2), an initial working assumption is that the scenarios in Table 7 will be used across all three models. We will revisit this assumption in the discussion of results.

Applying the three generic scenarios to each model of designing produces the datasets shown in Table 8. The number of steps for each model is calculated based on multiplying the number of elementary steps in each phase according to the number of iterations defined for the specific scenario. For example, DFSS-ICOV for scenario 1 has:

12 (0 iterations for the 12 elementary steps of phase 1)
+ 34 (1 iteration for the 17 elementary steps of phase 2)
+ 10 (1 iteration for the 5 elementary steps of phase 3)
+ 14 (1 iteration for the 7 elementary steps of phase 4)
= 70 steps in total.

Table 8 The number of steps produced by applying the three scenarios to each model of designing

	Systematic Approach	RUP	DFSS-ICOV
Scenario 1 (S1)	154	185	70
Scenario 2 (S2)	235	278	103
Scenario 3 (S3)	302	363	132

3.3 Quantitative Analysis

Having pre-processed the models of designing as sequences of steps, each of which produces an FBS design issue, allows applying cumulative occurrence analysis (Gero et al. 2014). This analysis has previously been applied to coded design protocols where designing is represented as a sequence of segments each producing one ontological design issue (Kannengiesser et al. 2013). The cumulative occurrence c of design issue x at design step n is defined as $c = \sum_{i=1}^{n} x_i$ where x_i equals 1 if design step i is coded as x and 0 if design step i is not coded as x. Plotting the results of this equation on a

graph with the design steps *n* on the horizontal axis and the cumulative occurrence *c* on the vertical axis will visualise the occurrence of the design issues. Figure 2 shows a general representation of such a graph.

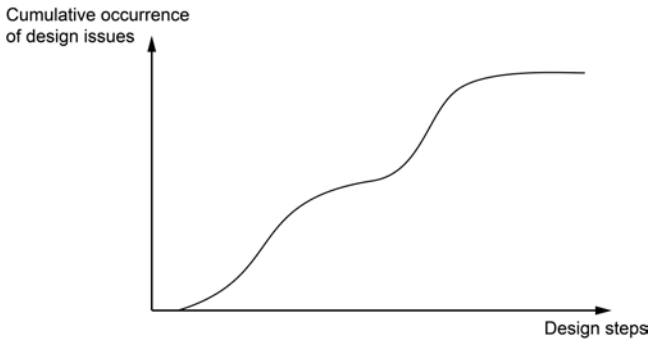

Fig. 2 Graphical representation of the cumulative occurrence of design issues across design steps

Drawing on Gero et al. (2014), four measures are used for analysing the cumulative occurrence-based representations of the different models of designing:

- *First occurrence at start*: This measure indicates whether design issues first occur near the start of designing or at a later stage.
- *Continuity*: This measure indicates whether design issues occur throughout designing or only up to a certain point.
- *Linearity*: This measure indicates whether the speed at which design issues are generated (or the cognitive effort expended on these design issues) is constant. It is measured using the coefficient of determination (R^2) that indicates linear fit and ranges from 0 to 1. As a threshold for linear fit we set the commonly used value of 0.95; i.e. if R^2 is at least 0.95, the graph is considered linear.
- *Slope*: The measure represents the rate at which design issues are generated. It is calculated only for graphs that are found to be linear.

The two measures of continuity and first occurrence at start are direct characterisations of cumulative occurrence, both available from a qualitative assessment of the graph. They can tell us whether a design issue is focused on from the very start of the design process and whether it is continuously focused on throughout the design activity, respectively. The measures of slope and linearity have been derived using a grounded theory approach, through a process of discovery by looking at the results of pre-

liminary simulation runs where some design issue graphs appeared to be linear.

All of these measures are independent of the number of design steps. This allows comparing models of designing that have different levels of detail and different numbers of iterations. We pose only one restriction on calculating slope and linearity to ensure sufficient statistical significance: that the number of occurrences per design issue is at least 10.

4. Simulation Results

In this Section we present the measures we derived from analysing the three models of designing. These measures are presented in Tables 9 to 14. In addition, to allow readers to carry out their own qualitative assessments, we also provide the raw data in the form of graphs representing the cumulative occurrence of design issues for scenario S2. These graphs are shown in Figures 3, 4 and 5. The vertical lines in these Figures separate the four phases in each model. They help in locating the occurrence of design issues within the respective model of designing, which is useful for deriving the measures of "first occurrence at start" and "continuity".

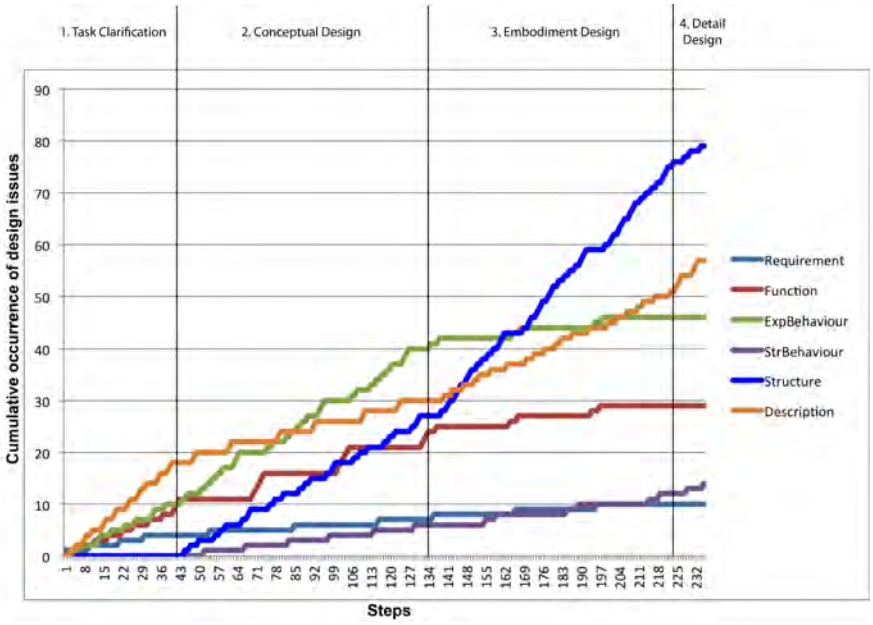

Fig3 Cumulative occurrence of design issues in the Systematic Approach (for the "typical" scenario S2)

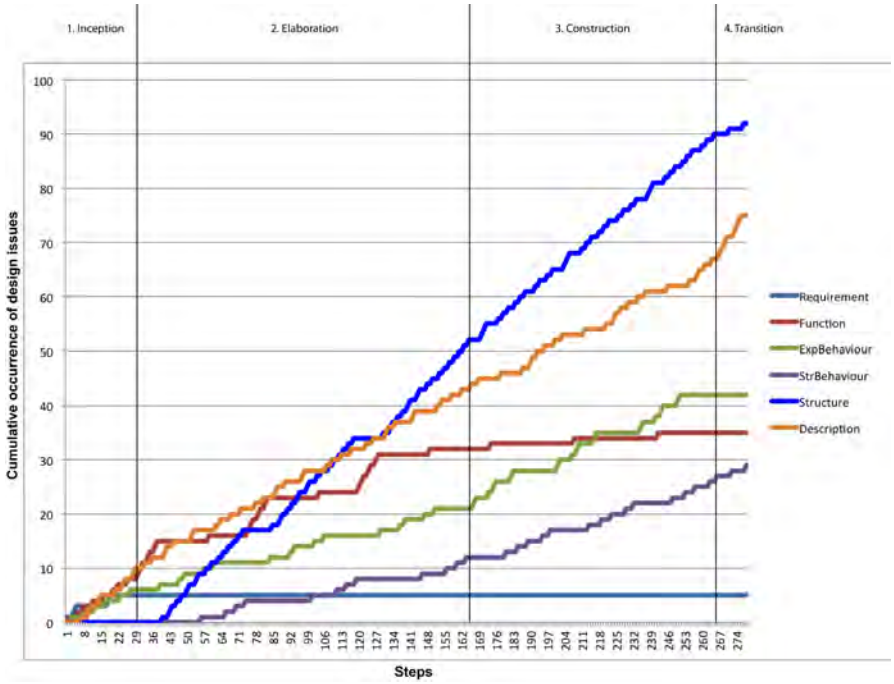

Fig4 Cumulative occurrence of design issues in the Rational Unified Process (for the "typical" scenario S2)

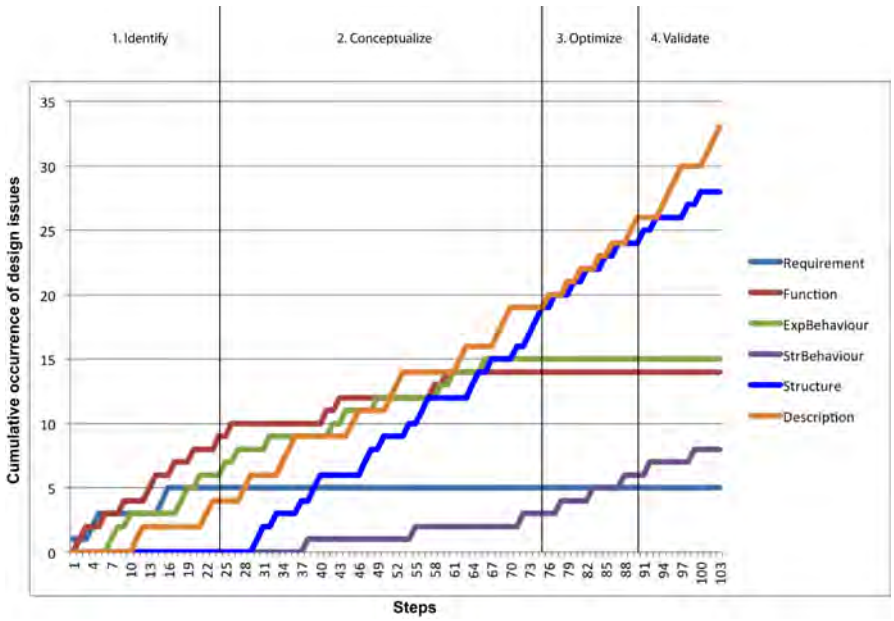

Fig5 Cumulative occurrence of design issues in DFSS-ICOV (for the "typical" scenario S2)

Table 9 Requirement issues

Model of design-ing	Slope	R^2	First occur-rence at start	Continuity	Linearity
Sys. App.					
S1*	---	---	Yes	No	---
S2	0.038	0.966	Yes	No	Yes
S3	0.036	0.977	Yes	No	Yes
RUP*					
S1, S2, S3	---	---	Yes	No	---
DFSS-ICOV*					
S1, S2, S3	---	---	Yes	No	---

* No statistical results produced due to small dataset (< 10 data points)

Table 10 Function issues

Model of design-ing	Slope	R^2	First occur-rence at start	Continuity	Linearity
Sys. App.					
S1	---	0.924	Yes	No	No
S2	---	0.947	Yes	No	No
S3	---	0.948	Yes	No	No
RUP					
S1	---	0.810	Yes	No	No
S2	---	0.860	Yes	No	No
S3	---	0.874	Yes	No	No
DFSS-ICOV					
S1*	---	---	Yes	No	---
S2	---	0.806	Yes	No	No
S3	---	0.820	Yes	No	No

* No statistical results produced due to small dataset (< 10 data points)

Table 11 Expected behaviour issues

Model of designing	Slope	R^2	First occurrence at start	Continuity	Linearity
Sys. App.					
S1	---	0.899	Yes	No	No
S2	---	0.924	Yes	No	No
S3	---	0.915	Yes	No	No
RUP					
S1	0.157	0.978	Yes	No	Yes
S2	0.154	0.981	Yes	No	Yes
S3	0.153	0.981	Yes	No	Yes
DFSS-ICOV					
S1*	---	---	Yes	No	---
S2	---	0.889	Yes	No	No
S3	---	0.888	Yes	No	No

* No statistical results produced due to small dataset (< 10 data points)

Table 12 Structure behaviour issues

Model of designing	Slope	R^2	First occurrence at start	Continuity	Linearity
Sys. App.					
S1**	0.065	0.959	No	Yes	Yes
S2**	0.065	0.982	No	Yes	Yes
S3**	0.065	0.989	No	Yes	Yes
RUP					
S1**	0.127	0.972	No	Yes	Yes
S2**	0.121	0.978	No	Yes	Yes
S3**	0.120	0.980	No	Yes	Yes
DFSS-ICOV					
S1*	---	---	No	Yes	---
S2*	---	---	No	Yes	---
S3**	---	0.919	No	Yes	No

* No statistical results produced due to small dataset (< 10 data points)

** The initial design steps of the protocol are ignored in slope and linearity calculations to take into account that the first occurrence is not at the start

Table 13 Structure issues

Model of designing	Slope	R^2	First occurrence at start	Continuity	Linearity
Sys. App.					
S1*	0.422	0.977	No	Yes	Yes
S2*	0.418	0.978	No	Yes	Yes
S3*	0.419	0.979	No	Yes	Yes
RUP					
S1*	0.394	0.997	No	Yes	Yes
S2*	0.390	0.999	No	Yes	Yes
S3*	0.386	0.999	No	Yes	Yes
DFSS-ICOV					
S1*	0.342	0.967	No	Yes	Yes
S2*	0.384	0.994	No	Yes	Yes
S3*	0.376	0.996	No	Yes	Yes

* The initial design steps of the protocol are ignored in slope and linearity calculations to take into account that the first occurrence is not at the start

Table 14 Description issues

Model of designing	Slope	R^2	First occurrence at start	Continuity	Linearity
Sys. App.					
S1	0.196	0.968	Yes	Yes	Yes
S2	0.198	0.972	Yes	Yes	Yes
S3	0.192	0.976	Yes	Yes	Yes
RUP					
S1*	0.251	0.991	Yes	Yes	Yes
S2*	0.241	0.996	Yes	Yes	Yes
S3*	0.243	0.997	Yes	Yes	Yes
DFSS-ICOV					
S1	0.340	0.979	Yes	Yes	Yes
S2	0.316	0.984	Yes	Yes	Yes
S3	0.325	0.984	Yes	Yes	Yes

* The initial design steps of the protocol are ignored in slope and linearity calculations to take into account that the first occurrence is not at the start

As a first observation, we note that there are no or only small differences among the three scenarios within each model of designing. All quali-

tative measures (first occurrence at start, continuity, and linearity) are the same for S1, S2 and S3 of each model. Differences in slope are not significant across the three scenarios.

When comparing the three models of designing with each other, we can make the following observations:

- *First occurrence at start:* In all three models, requirement issues, function issues, expected behaviour issues and description issues occur at the start (or in phase 1) of the design process. And in all three models, structure behaviour issues and structure issues occur later (in phase 2).
- *Continuity:* The cumulative occurrence of requirement issues, function issues and expected behaviour issues is discontinuous in all three models. Structure behaviour issues, structure issues and description issues are continuous in all three models.
- *Linearity:* The cumulative occurrence of function issues in all three models is non-linear, whereas the cumulative occurrence of structure issues and description issues in all three models is linear. For expected behaviour issues and structure behaviour issues, the results are inconsistent across the models.
- *Slope:* Slopes could not be compared for requirement issues (insufficient data), function issues (no linear graphs) and expected behaviour issues (linearity only for RUP). The slopes for other design issues were compared using one-way ANOVA tests, resulting in commonalities being found for neither structure behaviour issues ($F_{2,4} = 319.341$, $p < 0.05$), structure issues ($F_{2,6} = 11.889$, $p < 0.05$) nor description issues ($F_{2,6} = 220.841$, $p < 0.05$).

5. Discussion of Results

The results can be discussed in terms of the commonalities found across the three models of designing and in terms of the assumptions underlying the simulation models.

5.1 Identifying Commonalities across the Three Models of Designing

Our analysis has uncovered a number of commonalities among the three models of designing, independent of the number of iterations in each model (see Section 5.2 for an explanation of why they are independent). Table

15 summarises our findings, using "+" and "—" symbols to indicate the existence of a commonality: "+" indicates commonalities as specified on top of the Table, while "—" indicates commonalities as the negation of what is specified on top. These "negative commonalities" can only be derived from binary measures such as "first occurrence at start"; the negations of the other measures are too broad to for allow similar inverse conclusions. Empty spaces in the Table mean that there is no commonality.
Commonalities regarding the first occurrence of design issues near the start were found for all design issues. While requirements issues, function issues, expected behaviour issues and description issues occur near the start, structure behaviour issues and structure issues occur later. Commonalities regarding the continuity of the graph were found for structure behaviour issues, structure issues and description issues. The commonality of linearity was identified for structure issues and description issues. There are no commonalities regarding slope.

Table 15 Summary of commonalities

Design issue	First occurrence at start	Continuity	Linearity	Common Slope
Requirement	+			
Function	+			
Expected Behaviour	+			
Structure Behaviour	—	+		
Structure	—	+	+	
Description	+	+	+	

Some of the commonalities are consistent with the general goals of each of the four phases of the models, as introduced in Section 2. In the three models, requirement issues, function issues, expected behaviour issues and description issues start occurring in phase 1 as they are needed to define and document the design problem. The occurrence of these issues, except for description issues that continue to occur until the end, tends to diminish later as the focus of designing shifts towards possible design solutions. Structure issues and structure behaviour issues start occurring later, and continue to occur until the final design solution is determined, validated and documented. The existence of linearity of structure and description issues in all three models of designing is the most surprising outcome of

our analysis. It implies that during designing a uniform cognitive effort is expended on these issues, which has not been explicitly stated in previous work. Yet, the amount of cognitive effort differs as suggested by non-existence of a common slope for structure and description issues across different models of designing.

5.2 Revisiting Assumptions for the Simulation Models

The results of applying our approach shed some light on the validity of the assumptions used for constructing the simulation models (see Section 3.2).

Our first assumption was that every design step occurs only once within the same iteration. In common design practice this assumption is not realistic, because incomplete knowledge and design complexity often require repeating the same or similar design activities multiple times (Wynn et al. 2007). However, these task- and designer-specific variables cannot be taken into account for analysing models of designing that are independent of particular instances. Therefore the validity of the one-execution-per-step assumption must be based on its usefulness in analysing and comparing different models rather than its relation to the practice of designing. Increasing the number of executions per step, uniformly across all steps of a model, would not lead to changes in the four measures except for changed values for slopes. Even if the number of executions can vary for different steps in a model, only the shape of the graph would be affected in terms of its linearity or non-linearity, not a change from one shape to another. As a result, our assumption of one execution per step seems to be a useful and valid choice.

Our second assumption was related to the number of iterations of the different phases within a model of designing. We took Kruchten's (2004) three "typical" scenarios for RUP, each of which defines different numbers of iterations for the four phases, and applied them to the other models. The results show that the behaviour of the cumulative occurrence graphs in all three models of designing did not vary for the different scenarios. We might therefore simplify the assumption to include only one simple scenario where there are no iterations for any of the four phases. This would also facilitate the application of our approach to models that cannot be mapped onto the 4-phase process structure. For example, the VDI-2221 model (VDI 1985) has seven phases, and some variants of DFSS such as DMADV ("Define-Measure-Analyse-Design-Verify") and IDDOV ("Identify-Define-Design-Optimize-Validate") have five phases.

6. Conclusion

This paper proposed a quantitative approach for the analysis of domain-specific models of designing. Its application to three models of designing demonstrates its applicability to domains as different as engineering, software and service design. Based on its ontological foundations, the approach allows comparisons between models from different design domains. The comparison of the models analysed in this paper shows that there are some strong commonalities that provide support for the hypothesis that designing is an act that is independent of the domain of its application. This may have important implications for design education: If designing is foundational and is shown to be domain-independent and different to science and humanities, then consideration should be given to teaching design in parallel with science and humanities.

The work presented in this paper should be regarded as preliminary, both in terms of the method used and the findings regarding commonalities across design domains. The method presented is limited to the descriptive capacities of the FBS ontology. We have used this ontology since many researchers have already used it as a coding scheme for representing design processes in various domains. Other design ontologies could be used for coding models of designing if they have the capacity to provide a coding scheme for design processes. However, to the best of our knowledge, no other ontological coding scheme exists with similar domain-independence as the FBS coding scheme. Further research in the analysis method is therefore more likely to focus on the particular measures used, including their definition and completeness. Graphs other than those based on cumulative occurrence may be used with different associated measures.

The findings of our comparisons of engineering, software and service design are limited by the choice of models of designing. We chose the Systematic Approach, RUP and DFSS-ICOV because of their popularity and their detailed level of description. Yet, it remains to be tested if analysing other models in the same domain produce similar results. A difficulty here is that many domain-specific models of designing are quite coarse-grained and cannot be easily represented in sufficient detail as needed for the proposed simulation model and associated statistical analyses.

The results presented in this paper provide a starting point for future research. For example, they could be compared with empirical research, as there are many protocol studies available using the same FBS design issue scheme. Such comparisons would provide the basis to examine differences between models of designing and designing as practised.

Acknowledgements

This research is supported in part by grants from the US National Science Foundation grant nos. IIS-1002079, CMMI-1400466 and CMMI-1161715. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Asimow M (1962) *Introduction to Design*, Prentice-Hall, Englewood Cliffs, NJ.

Boon M and Knuuttila T (2009) Models as epistemic tools in engineering sciences, *in* Meijers (ed.) *Philosophy of Technology and Engineering Sciences*, Elsevier, Amsterdam, pp. 693-726.

Chakrabarti A and Blessing LTM (2014) Theories and models of design: A summary of findings, *in* A Chakrabarti and LTM Blessing (eds) *An Anthology of Theories and Models of Design*, Springer, London, pp. 1-46.

Chittaro L and Kumar AN (1998) Reasoning about function and its applications to engineering, *Artificial Intelligence in Engineering* **12**(4): 331-336.

Cross N (1982) Designerly ways of knowing, *Design Studies* **3**(4): 221-227.

Dym C (1994) *Engineering Design: A Synthesis of Views*, Cambridge University Press, Cambridge, MA.

Eder WE (2012) *Comparison of several design theories and methods with the legacy of Vladimir Hubka*, Public Report, The Design Society.

El-Haik B and Roy DM (2005) *Service Design for Six Sigma: A Roadmap for Excellence*, John Wiley & Sons, Hoboken, NJ.

Frey DD and Dym CL (2006) Validation of design methods: lessons from medicine, *Research in Engineering Design* **17**(1): 45-57.

Gero JS (1990) Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26-36.

Gero JS and Kannengiesser U (2004) The situated function-behaviour-structure framework, *Design Studies* **25**(4): 373-391.

Gero JS, Kannengiesser U and Pourmohamadi M (2014) Commonalities across designing: Empirical results, *in* JS Gero (ed), *Design Computing and Cognition'12*, Springer, pp. 285-302.

Gero JS and McNeill T (1998) An approach to the analysis of design protocols, *Design Studies* **19**(1): 21-61.

Grabowski H, Rude S and Grein G (1998) *Universal Design Theory*, Shaker Verlag, Aachen.

Hubka V and Eder WE (1996) *Design science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*, Springer, London.

Kan J and Gero JS (2005) Design behaviour measurement by quantifying linkography in protocol studies of designing, *in* JS Gero and U Lindemann

(eds) *Human Behaviour in Designing'05,* Key Centre of Design Computing and Cognition, University of Sydney, Australia, pp. 47-58.

Kannengiesser U, Williams C. and Gero JS (2013) What do the concept generation techniques of TRIZ, morphological analysis and brainstorming have in common?, *DS 75-7: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol.7: Human Behaviour in Design*, Seoul, Korea.

Kruchten P (2004) *The Rational Unified Process: An Introduction*, Addison-Wesley, Upper Saddle River, NJ.

Lawson B (1980) *How Designers Think: The Design Process Demystified*, Architectural Press, Amsterdam.

Lindemann U (2014) Models of design, *in* A Chakrabarti and LTM Blessing (eds) *An Anthology of Theories and Models of Design*, Springer, London, pp. 121-132.

Pahl G and Beitz W (2007) *Engineering Design: A Systematic Approach*, Springer, Berlin.

Roozenburg NFM and Cross NG (1991) Models of the design process: Integrating across the disciplines, *Design Studies* **12**(4): 215-220.

Sim SK and Duffy AHB (2003) Towards an ontology of generic engineering design activities, *Research in Engineering Design* **14**(4): 200-223.

Tate D and Nordlund M (1996) A design process roadmap as a general tool for structuring and supporting design activities, *Proceedings of the Second World Conference on Integrated Design and Process Technology (IDPT-Vol. 3)*, Society for Design and Process Science, Austin, TX, pp. 97-104.

Ullman DG, Dietterich TG and Stauffer LA (1988) A model of the mechanical design process based on empirical data, *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing* **2**(1): 33-52.

Unger D and Eppinger S (2011) Improving product development process design: a method for managing information flows, risks, and iterations, *Journal of Engineering Design* **22**(10): 689-699.

VDI (1985) *VDI-Richtlinie 2221 (Entwurf): Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte*, VDI-Verlag, Düsseldorf.

Vermaas PE (2009) The flexible meaning of function in engineering, in M Norell Bergendahl et al. (eds) *Proceedings of the 17th International Conference on Engineering Design (ICED'09), Vol. 2*, The Design Society, pp. 113-124.

Vermaas PE (2014) Design theories, models and their testing: On the scientific status of design research, *in* A Chakrabarti and LTM Blessing (eds) *An Anthology of Theories and Models of Design*, Springer, London, pp. 47-66.

Visser W (2009) Design: one, but in different forms, *Design Studies* **30**(3): 187-223.

Welch RV and Dixon JR (1994) Guiding conceptual design through behavioral reasoning, *Research in Engineering Design* **6**(3): 169-188.

Wynn DC, Eckert CM and Clarkson PJ (2007) Modelling iteration in engineering design, *International Conference on Engineering Design (ICED'07)*, Paris, France, pp. 561/1-11.

Yang K and El-Haik B (2003) *Design for Six Sigma: A Roadmap for Product De-*

velopment, McGraw-Hill, New York.

Appendix A – The Situated FBS Framework

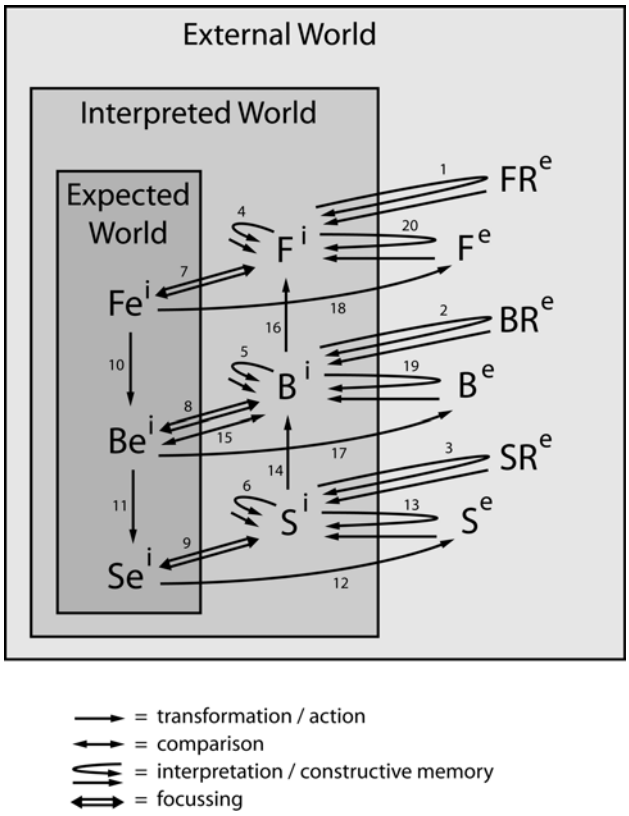

Fig. A1 The situated FBS framework

The *external world* contains objects and representations in the environment of the designer.

The *interpreted world* contains experiences, percepts and concepts produced by the designer's interactions with the external world.

The *expected world* contains the designer's hypotheses, goals and expected results of actions.

Explanation of symbols:

Fe^i: Expected Function
F^i: Interpreted Function
F^e: External Function
FR^e: Requirement on Function

Be^i: Expected Behaviour

B^i: Interpreted Behaviour
B^e: External Behaviour
BR^e: Requirement on Behaviour

Se^i: Expected Structure
S^i: Interpreted Structure
S^e: External Structure
SR^e: Requirement on Structure

Appendix B - Pahl and Beitz' Systematic Approach

Table B1. Phase 1: Task Clarification (numbers refer to sFBS process labels; page numbers refer to Pahl and Beitz (2007))

Activity	Design issue	sFBS step	Comments
1.1 Define basic market demands	R	1	-a "task description" is given in the external world, describing the product's desired "functionality and performance" (p. 145)
	F, Be	4, 5	-generated internally: "Basic requirements are always implicit requirements, i.e. they are not articulated by the customer." (p. 150) -can be related to F and B in new product development
	D, D	18, 17	-all requirements are compiled as a requirements list that is produced in the external world
1.2 Define attractiveness demands of the market segment	F, Be	4, 5	-generated internally: "Attractiveness requirements are again implicit requirements." (p. 151) -can be related to F and B in new product development
	D, D	18, 17	-all requirements are compiled as a requirements list that is produced in the external world
1.3 Document customer-specific technical performance requirements	R	2	-given in the external world: "Technical performance requirements are explicit requirements. They are articulated by the customer and can usually be specified precisely." (pp. 150) -"performance" corresponds to B; see also the examples on p. 151

	D	17	-all requirements are compiled as a requirements list that is produced in the external world
1.4 Refine and extend the requirements using the checklist and scenario planning	F, Be, Be D, D	4, 5, 10 18, 17	-these requirements are internally developed; no specific guidance from external world -all requirements are compiled as a requirements list that is produced in the external world
1.5 Determine demands and wishes	F, Be D, D	7, 8 18, 17	-distinction between demands and wishes is about formulating a design state space (focussing) -demands/wishes distinction is included in the requirements list in the external world: see example on p. 154

Table B2. Phase 2: Conceptual Design (numbers refer to sFBS process labels; page numbers refer to Pahl and Beitz (2007))

Activity	**Design issue**	**sFBS step**	**Comments**
2.1 Abstract to identify the essential problems	F	20	-related to F based on the external requirements list: "Here the task is to analyse the requirements list with respect to the required function and essential constraints in order to confirm and refine the crux of the problem." (p. 164) "[…] the final formulation can be derived in a way that does not prejudice the solution, i.e. is solution-neutral, and at the same time turns it into a function." (p. 165)
2.2 Establish function structures: overall function – subfunctions	F F	4 7	-internal generation of new (sub-) functions -prioritization of functions: "It is useful to start by determining the main flow in a technical system […]. The auxiliary flows should only be considered later." (p. 171) "The search for solutions […] then focuses on the subfunctions that are essential for the solution and on which the solutions of other subfunctions depend […]." (p. 181)
2.3 Search for working principles that fulfil the subfunctions			-"Only the combination of the physical effect with the geometric and material characteristics […] allows the principle of the solution to emerge. This interrela-

	Be	10	tionship is called the working principle […].” (p. 40)
	S	6	-involves generating physical effects (B) based on subfunctions (F)
	Be, S, D, D, S, Be, Bs, Be	8, 9, 12, 17, 13, 19, 14, 5	-involves generating working surfaces (S) and types of materials (S), both can be expressed as S variables -may involve incrementally focusing on B and S, producing, interpreting and analysing external S: “[…] the stepwise generation of working principles, through the search for physical effects and the subsequent form design features, is often integrated mentally by producing sketches of solutions. This is because designers think more in configurations and representation of principles than in physical equations.” (p. 189)
	R	19, 3	-involves interpreting relevant behaviours in the requirements list (B) and, if available, structure (SR^e): “[Extensive solution fields] should be reduced as soon as feasible working principles emerge by checking against the demands in the requirements list.” (p. 189)
2.4 Combine working principles into working structures	Be	10	-“The combination of several working principles results in the working structure of a solution.” (p. 40) -involves creating sets of working principles “to fulfil the overall function” (p. 184)
	S	6	-involves creating sets of working surfaces (S) and types of materials (S)
2.5 Select suitable combinations	Be, S	8, 9	-“selection” here corresponds to focussing on B and S
2.6 Firm up into principle solution variants	Be, S	10, 11	-involves generating more information about the working principles, through additional variables and some values for B and S: “The most important properties of the proposed combination of principles must first be given a much more concrete qualitative, and often also a rough quantitative, definition.” (p. 190) “The fulfilment of the technical function alone does not complete the task of designers […]. […] In addition, the solu-

			tion of technical tasks imposes certain constraints or requirements resulting from ergonomics, production methods, transport facilities, the intended operation, etc. […].” (p. 43) “It is advisable to consider these guidelines [a list of general constraints] even during the conceptual phase.” (p. 44)
	D, D	12, 17	-produces models and sketches (S) and calculations and tests/simulations (B) in the external world (p. 190)
2.7 Evaluate variants against technical and economic criteria	Be	19	-“Identifying evaluation criteria” (p. 192) involves interpretation of external B: “This step is based, first of all, on the requirements list.” (p. 192)
	Be, Be	5, 8	-involves generating and focussing on additional B using general checklist (p. 193)
	Be	8	-“Weighting the evaluation criteria” (p. 194) corresponds to focussing on B
	S	13	-“Compiling parameters” (p. 194) involves gathering data from the results of step 2.6: “Whatever quantitative information is available at this stage should also be included. Such quantitative data generally result from the step we have called “firming up into principle solution variants”.” (p. 194)
	Bs	14	-“Assessing values” uses ratings such as “the 0-4 scale proposed in VDI Guideline 2225” (p. 195); “Determining overall value” is “a matter of simple addition” (p. 197)
	--	15	-“Comparing concept variants” (p. 197)
	S, S, F, F	6, 9, 16, 7	-may involve generating and focussing on new solutions, through “transfer of better subsolutions from other variants” (p.198), and deriving and focussing on additional functions through fault-tree analysis (see example in Fig. 10.7, p. 525)

Table B3. Phase 3: Embodiment Design (numbers refer to sFBS process labels; page numbers refer to Pahl and Beitz (2007))

Activity	Design	sFBS	Comments

	issue	step	
3.1 Identify embodiment-determining requirements	F, Be, R, F, Be, S	20, 19, 3, 7, 8, 9	-involves interpretation of the requirements list in the external world, and selection (focussing): "Starting with the principle solution, and using the requirements list, the first step is to identify those requirements that have a crucial bearing on the embodiment design." (p. 228)
3.2 Produce scale drawings of spatial constraints	D	12	-involves creation of external S
3.3 Identify embodiment-determining main function carriers	S	9	-involves selection (focussing) of S in terms of "the overall embodiment-determining main function carriers" (p. 228)
3.4 Develop preliminary layouts and form designs for the embodiment-determining main function carriers	S D	11 12	-involves developing S in terms of the "general arrangement, component shapes and materials" (p. 230) -involves producing external representations of S: "The representation of the spatial constraints and the embodiment is now generally obtained by creating a full 3-D digital model." (p. 231)
3.5 Select suitable preliminary layouts	S	11	-this selection is part of the B-to-S transformation
3.6 Develop preliminary layouts and form designs for the remaining main function carriers	S S D	9 11 12	-involves focussing on "the remaining main function carriers" (p. 230) -involves developing S for these function carriers -involves producing external representations of S: "The representation of the spatial constraints and the embodiment is now generally obtained by creating a full 3-D digital model." (p. 231)
3.7 Search for solutions to auxiliary functions	S S	9 6	-involves focussing in terms of selecting, "where possible, […] known solutions." (p. 230) -may involve generating "special solutions, using the procedures already described in Section 3.2 [including creativity techniques such as brainstorming, synectics etc.] and Chapter 6." (p. 230)
3.8 Develop de-	S	11	-involves developing S for the main

tailed layouts and form designs for the main function carriers ensuring compatibility with the auxiliary function carriers	D	12	function carriers -involves producing external representations of S: "The representation of the spatial constraints and the embodiment is now generally obtained by creating a full 3-D digital model." (p. 231)
3.9 Develop detailed layouts and form designs for the auxiliary function carriers and complete the overall layouts	S D	11 12	-involves developing S for the auxiliary function carriers -involves producing external representations of S: "The representation of the spatial constraints and the embodiment is now generally obtained by creating a full 3-D digital model." (p. 231)
3.10 Evaluate against technical and economic criteria	S, Bs	13, 14, 15	-must involve interpretation of the external model, and analysis and comparison
3.11 Optimise and complete form designs	S D	11 12	-involves changing S "by eliminating the weak spots" (p. 231) -involves producing external representations of S: "The representation of the spatial constraints and the embodiment is now generally obtained by creating a full 3-D digital model." (p. 231)
3.12 Check for errors and disturbing factors	S, Bs S, S, S	13, 14, 15 13, 6, 9	-must involve interpretation of the external model, and analysis and comparison -may involve generating and focussing on new S as a result of fault-tree analysis (p. 526)
3.13 Prepare preliminary parts lists and production documents	D	12	-creates documentation (S) in the external world

Table B4. Phase 4: Detail Design (numbers refer to sFBS process labels; page numbers refer to Pahl and Beitz (2007))

Activity	**Design issue**	**sFBS step**	**Comments**
4.1 Finalise details; complete detail drawings	S	11	-involves optimisation of S by selecting "the most suitable materials […], at cost-effectiveness and at ease of production, with due attention being paid to standards […]." (p. 437)

	D	12	-involves generating S in the external world, "comprising the detailed drawing of components, and the detailed optimisation of shapes, materials, surfaces, tolerances and fits." (p. 437)
4.2 Integrate into overall layout drawings, assembly drawings and parts lists	D	12	-involves a re-representation of external S
4.3 Complete production documents with production, assembly, transport and operating instructions	D	12	-involves a re-representation of external S
4.4 Check all documents for standards, completeness and correctness	S, Bs	13, 14, 15	-must involve interpretation of the external documents, and analysis and comparison

Appendix C - Rational Unified Process

Table C1. Phase 1: Inception (numbers refer to sFBS process labels; page numbers refer to Kruchten (2004))

Activity	Design issue	sFBS step	Comments
1.1 Analyze the problem	R F, Be	1 4, 5	-we assume some expression of a need to initiate designing -involves generating internal F and B, through "gain[ing] agreement on a statement of the problem we are trying to solve" (p. 164) and "identify[ing] the boundaries and constraints of the system" (p. 164)
1.2 Understand stakeholder needs	R, R	1, 2	-involves eliciting external requirements on F and B, through "gather[ing] stakeholder requests and […] obtaining a clear understanding of the real needs of the users and stakeholders of the system" (p. 166)
1.3 Define the system	F	4	-involves generating F by "establish[ing] the set of system features to be consid-

	D	18	ered for delivery" (p. 166) -involves producing external F "to set realistic expectations with the stakeholders on what features will be delivered" (p. 166)
1.4 Manage the scope of the system	F, Be D, D	7, 8 18, 17	-involves selecting or focussing on expected F and B -involves producing external F and B as "requirements attributes" (p. 166)
1.5 Refine the system definition	F, Be D, D	4, 10 18, 17	-involves generating F and B, through establishing "the functionality of the system […] and other important requirements, such as nonfunctional requirements, design constraints, and so forth" (p. 166) -involves producing external F and B, "to come to an agreement with the customer" (p. 166)

Table C2. Phase 2: Elaboration (numbers refer to sFBS process labels; page numbers refer to Kruchten (2004))

Activity	**Design issue**	**sFBS step**	**Comments**
2.1 Decide which use cases and scenarios will drive the development of the architecture	F F	20 7	-involves interpreting external F, by "discussing an initial use-case view" (p. 251) -involves "determin[ing] which use cases and scenarios should be focused on in this iteration" (p. 251)
2.2 Understand this driver in detail and inspect the results	F D	4 18	-involves detailing and "restructur[ing] the use-case model as a whole" -involves producing external F as "use-case model and supplementary specification" to be "reviewed and approved" (p. 251)
2.3 Reconsider use cases and risks	F F	20 7	-involves interpreting external F by "revisit[ing] the use-case view" -involves focussing by "select[ing] the set of use cases and scenarios to be analyzed, designed, and implemented in the current iteration." (p. 251)
2.4 Prototype the user interface	D	18	-involves producing external F by "build[ing] a user-interface prototype to get feedback from prospective users" (p. 252);

	F, F	20, 7	-we presume that this feedback may lead to a reformulation of F
2.5 Find obvious classes, do initial subsystem partitioning, and look at use cases in detail	Be S D, D	10 6 17, 12	-involves generating expected B based on expected F by "identif[ying] the analysis mechanisms that constitute common solutions to common problems during analysis" (p. 252) -involves generating S by "start[ing] finding classes or objects for this iteration's use cases or scenarios" (p. 252) -involves producing external B and S as a "software architecture document" (p. 252)
2.6 Refine and homogenize classes and identify architecturally significant ones; inspect results	S S D	11 9 12	-involves synthesizing S by "refin[ing] the classes identified" (p. 252) -involves focussing on S by "identif[ying] a number of classes that should be considered architecturally significant" (p. 252) -involves producing external S by "includ[ing the architecturally significant classes] in the logical view (Artifact: Software Architecture Document)" (p. 252)
2.7 Consider the low-level package partitioning	S	11	-involves synthesizing S by "organiz[ing] some of the classes into design packages" (p. 252)
2.8 Adjust to the implementation environment, decide the design of the key scenarios, and define formal class interfaces; inspect results	Be, S Be, S S D, D	5, 6 8, 9 11 17, 12	-involves generating B and S as constraints imposed by "the implementation environment" (p. 253) -involves focusing on B and S to provide "detailed requirements that are then put on each object" (p. 253) -involves synthesizing S by "merg[ing the detailed requirements] into consistent and formal interfaces on their classes" (p. 253) -involves producing external B and S by "updat[ing] the logical view accordingly" (p. 253)
2.9 Consider concurrency and distribution of the architecture	S	11	-involves synthesizing S based on "the collaborating objects in interaction diagrams" (p. 253)
2.10 Inspect the	S, Bs	13,	-likely to involve interpreting external

architectural design		14, 15	S, and deriving and evaluating B
2.11 Consider the physical packaging of the architecture	Be, S	5, 6	-involves generating B and S by "defin[ing] the implementation view" (p. 253)
2.12 Plan the integration	F S	7 9	-involves focussing on F by "stud[ying] the use cases that are to be implemented in this iteration" (p. 253) -involves focussing on S by "defin[ing] the order in which subsystems should be implemented" (p. 253)
2.13 Plan integration tests and system tests	Be D	10 17	-involves generating expected B based on expected F by "plan[ning] the system tests and the integration tests, selecting measurable testing goals [which] could be expressed in terms of the ability to execute a use-case scenario with a certain response time or under specified load" (p. 253) -involves producing external B as a "test plan" (p. 254)
2.14 Implement the classes and integrate	S, D, S, Bs	11,12, 13,14, 15	-involves synthesizing and externalizing S, then interpreting it and deriving and evaluating B, by "cod[ing] and unit-test[ing] the classes identified in the architectural design" (p. 254)
2.15 Integrate the implemented parts	S, D, S, Bs	11,12, 13,14, 15	-involves synthesizing and externalizing S, then interpreting it and deriving and evaluating B, by "integrat[ing] the subsystems into an executable architectural prototype [and then testing it]" (p. 254)
2.16 Assess the executable architecture	S, D, S, Bs	11,12, 13,14, 15	-involves synthesizing and externalizing S, then interpreting it and deriving and evaluating B, as "[o]nce the whole system […] has been integrated, the System Tester tests the system" (p. 254)

Table C3. Phase 3: Construction (numbers refer to sFBS process labels; page numbers refer to Kruchten (2004))

Activity	**Design issue**	**sFBS step**	**Comments**
3.1 Plan system-level integration	S	9	-involves focussing on S by selecting "the order in which subsystems are to be

	D	12	put together to form a working and testable configuration" (p. 255) -involves producing external S as "documented in the Build Plan" p. 256)
3.2 Plan and design system-level test	Be Be D	10 5 17	-involves generating expected B, presumably based on expected F (the use-case scenarios) -involves generating B from "preceding iterations, which could be modified to be reused" (p. 256) -involves producing external B as "test scripts" (p. 256)
3.3 Refine use-case realizations	S S S	11 6 9	-involves synthesizing S by "refin[ing] the classes identified in previous iterations" (p. 256) -may involve generating S, as "[c]lasses may need to be added" (p. 256) -may involve focussing on S, as "[c]hanges to classes may require a change in subsystem partitioning" (p. 256)
3.4 Plan and design integration tests at the subsystem and system levels	Be F, Be Be	19 7, 8 10	-involves interpreting external B as the "Test Plan" (p. 256) -involves focussing on F and B: "The Designer identifies the functionality that will be tested together and the stubs and drivers that must be developed to support the integration tests" (p. 256) -involves generating B, presumably based on expected F and B ("based on the input from the Test Designer", p. 256) by "develop[ing] the stubs and drivers" (p. 256)
3.5 Develop code and test unit	S, D, S, Bs S, Be	11,12, 13,14, 15 9, 8	-involves synthesizing and externalizing S, then interpreting it and deriving and evaluating B, by "implement[ing] the classes in the Implementation Model [and fixing] defects" (p. 256) -may involve reformulation of S and B in terms of "design changes based on discoveries made in implementation" (p. 256)
3.6 Plan and implement unit test	Be	10	-involves generating expected B, presumably based on expected F
3.7 Test unit with-	S, Bs	13,14,	-involves interpreting external S and

in a subsystem		15	deriving and evaluating B
3.8 Integrate a subsystem	S, D	11, 12	-involves synthesizing S and producing external S by "bringing together completed and stubbed classes that constitute a build" (p. 257)
3.9 Test a subsystem	S, Bs D	13,14, 15 17	-involves interpreting external S and deriving and evaluating B -involves producing external B by "log[ging] the defects for arbitration to decide when they are to be fixed" (p. 257)
3.10 Release a subsystem	D	12	-involves producing external S by "releas[ing] the tested version of the subsystem […] into an area where it becomes visible, and usable, for system-level integration" (p. 257)
3.11 Integrate the system	S, D	11, 12	-involves synthesizing S and producing external S by "add[ing] subsystems and creat[ing] a build that is handed over to the Integration Testers" (p. 257)
3.12 Test integration	S, Bs D	13,14, 15 17	-involves interpreting external S and deriving and evaluating B -involves producing external B by "log[ging] the defects" (p. 257)
3.13 Test the system	S, Bs	13,14, 15	-involves interpreting external S and deriving and evaluating B

Table C4. Phase 4: Transition (numbers refer to sFBS process labels; page numbers refer to Kruchten (2004))

Activity	**Design issue**	**sFBS step**	**Comments**
4.1 Plan deployment	D	17	-involves producing external B as the "beta test program" (p. 242)
4.2 Develop support material	D, D	18, 17	-involves producing external F and B by providing "information that will be required by the end user to install, operate, use, and maintain the delivered system" (p. 242)
4.3 Produce deployment unit	D	12	-involves producing external S as the final software (p. 242)
4.4 Beta test product	S, Bs	13,14, 15	-involves interpreting external S and deriving and evaluating B

Appendix D - DFSS-ICOV

Table D1. Phase 1: Identify (numbers refer to sFBS process labels; page numbers refer to El-Haik and Roy (2005))

Activity	Design issue	sFBS step	Comments
1.1 Idea creation	R	1	-we assume some expression of a need to initiate designing
	F, F	4, 7	-involves generating and focussing on F, by creating "a market vision, with an assessment of marketplace advantages" (p. 83)
1.2 Voice of the customer and business	R, R	1, 2	-involves interpreting F and B, by "obtain[ing] customer needs and wants (p. 84)
	F, Be	4, 5	-involves generating F and B, by "identify[ing] and fill[ing] gaps in customer-provided requirements", by "establish[ing] metrics for CTSs", "quantify[ing] CTSs" and by "align[ing] with business objectives" (p. 84)
	Be	10	-involves deriving B from F, by "translat[ing] the VOC to CTSs" (p. 84)
	F, Be	7, 8	-involves focussing on F and B, by "conduct[ing] risk assessment" (p. 84) and performing a Kano analysis (p. 118)
	D, D	18,17	-involves externalizing F and B, by producing "a list of the voice of the customer (VOC)" (p. 84) and a HOQ 1 (p. 386)

Table D2. Phase 2: Conceptualize (numbers refer to sFBS process labels; page numbers refer to El-Haik and Roy (2005))

Activity	Design issue	sFBS step	Comments
2.1 Concept development	F, Be	16, 5	-involves deriving F from B, and generating B by "translat[ing] customer requirements (CTSs or big *Y*s) to service/process functional requirements" (p. 86)
	F, Be	7, 8	-involves focussing on F and B, by determining "prioritized functional requirements" (p. 114)
	D, D	18,17	-involves externalizing F and B, by pro-

			ducing a HOQ 2 (p. 121)
	S	6	-involves generating S, by "generat[ing] design alternatives" (p. 87)
	S	9	-involves focussing on S, by "select[ing] a process or service conceptual design" (p. 87) using the Pugh selection method
2.2 Preliminary design	Be, S	10,11	-involves "determin[ing] a set of design parameters which will fulfill the FRs" (p. 122)
	D, D, D	18,17, 12	-involves externalizing F, B and S, by producing a HOQ 3 (p. 127)
	S, Bs	13,14 15	-involves interpreting S and deriving and evaluating B, by identifying the potential failure modes and effects within an FMEA (p. 249)
	S, S	6, 9	-involves generating and focussing on S, by "decid[ing] on design controls" (p. 254)

Table D3. Phase 3: Optimize (numbers refer to sFBS process labels; page numbers refer to El-Haik and Roy (2005))

Activity	Design issue	sFBS step	Comments
3.1 Design optimization			One of the typical tools for this phase is suggested to be Design of Experiments (DOE) (p. 90)
	S, D	11,12	-involves synthesizing and externalizing S, by "vary[ing] the factors that can cause a change in the performance of *y*" (p. 264) and conducting experiments
	S, Bs	13,14 15	-involves interpreting S, and deriving and evaluating B, by collecting and analysing data from the experiments (p. 277)
	D	17	-involves externalizing B, by using design scorecards (p. 90)

Table D4. Phase 4: Validate (numbers refer to sFBS process labels; page numbers refer to El-Haik and Roy (2005))

Activity	Design issue	sFBS step	Comments
4.1 Verification	D	12	-involves externalizing S, by executing "pilot tests" (p. 91)
	S, Bs	13,14	-involves interpreting S and deriving and

	S	15 11	evaluating B, by identifying the potential failure modes and effects within an FMEA (p. 91) -involves synthesizing S, by "refining" the service (p. 91)
4.2 Launch readi-ness	D, D, D	18,17 12	-involves externalizing F, B and S, by "change management message[s]" (p. 406), "process capability modeling" (p. 92) and "mistake proofing" (p. 92)