What distinguishes a model of systems engineering from other models of designing? An ontological, data-driven analysis

Udo Kannengiesser¹ · John S. Gero²

Received: 4 February 2021 / Revised: 7 December 2021 / Accepted: 13 December 2021 / Published online: 15 January 2022
© The Author(s) 2022

Abstract

This paper investigates how the core technical processes of the INCOSE model of systems engineering differ from other models of designing used in the domains of mechanical engineering, software engineering and service design. The study is based on fine-grained datasets produced using mappings of the different models onto the function-behaviour-structure (FBS) ontology. By representing every model uniformly, the same statistical analyses can be carried out independently of the domain of the model. Results of correspondence analysis, cumulative occurrence analysis and Markov model analysis show that the INCOSE model differs from the other models in its increased emphasis on requirements and on behaviours derived from structure, in the uniqueness of its verification and validation phases, and in some patterns related to the temporal development and frequency distributions of FBS design issues.

Keywords Systems Engineering · Mechanical Engineering · Software Engineering · Service Design · Function-Behaviour-Structure (FBS) Ontology

1 Introduction

The analysis and design of systems, which are the subject of systems thinking and systems engineering, have a long tradition in research and practice. More recently, this field of science has gained renewed attention, driven by technological advances leading to complex, engineered systems such as cyber-physical systems (CPS). Generally, the notion of a system is increasingly used when objects or processes are examined with respect to their interactions with other objects or processes in their natural or socio-technical environment. A system is viewed as a set of elements that interact in a way to realise a defined objective (von Bertalanffy 1968). A system may itself be an element of a larger system. Thus, a system may encompass several hierarchical levels. Systems composed of large numbers of hierarchical levels are commonly referred to as systems of systems (Maier 1998).

System hierarchies provide the basis for systems thinking. Specifically, systems thinking has two aspects (Kasser

2020): systemic and systematic thinking. Systemic thinking is generally viewed as the opposite of reductionism: Rather than reducing a system into a set of components that are studied in isolation, systemic thinking looks at the system as a whole, in terms of the interactions of components and their emergent effects. This often leads to viewing a system as a subsystem of a larger system, thus allowing to understand or explain it based on its role within that system. This “bigger picture” can potentially reveal new insights, which may open up new ways of framing analysis or design issues for the system.

Systematic thinking aims to break a complex issue down into smaller issues to be addressed in a methodical, stepwise manner. In this respect, this type of thinking can be viewed as reductionist. On the other hand, in systematic thinking the results obtained at the component level are integrated to form the overall solution to the original problem.

Systems engineering is closely connected to systematic thinking, as it follows the basic decomposition—recomposition approach: requirements are broken down at consecutive levels in the system hierarchy and transformed into design components. These components are then composed into subsystems and finally the overall system. The effects of the composition, at every system level, are validated and verified. Such a systematic approach aims to reduce the

✉ Udo Kannengiesser
udo.kannengiesser@jku.at

¹ Johannes Kepler University Linz, Linz, Austria

² University of North Carolina at Charlotte, Charlotte, USA

likelihood of errors and flaws in the system's design and is often used for systems that are safety-critical or have long life cycles.

Engineering design and systems engineering are seen to share similar goals and methods (Finkelstein et al. 1989). This is not surprising, since systems engineering approaches were originally developed by (mostly mechanical and electrical) engineers in military and aeronautics (Chang et al. 2008; Honour 2018). Systems are seen as particular types of products that are large and complex (Finkelstein et al. 1989). Therefore, one may view systems engineering as a subdiscipline of engineering design. On the other hand, systems often involve heterogeneous technologies that require expertise from multiple engineering disciplines. These disciplines “fall into categories in descending degrees of abstraction beneath [systems engineering]” (Finkelstein et al. 1989). For example, mechatronic systems engineering encompasses mechanical engineering, electrical engineering and information technology as subdisciplines (VDI 2004). Systems engineering is viewed as a high-level framework or “macro-level” model (Wynn and Clarkson 2018) that includes project structures and contextual aspects such as organisational and managerial issues.

While the beginnings of the systems engineering field were mainly influenced by traditional engineering design, in the 1990s there was a significant shift towards adopting concepts from software engineering (Honour 2018). This was a result of the increasing importance of software being embedded in technical systems throughout the 1980s (Godfrey 1990). A number of international standards such as ISO/IEC/IEEE (2011, 2015, 2018) were defined aiming at the uniform development of systems and software using the same models and methods.

A more recent trend in systems engineering is the addition of services to result in product-service systems (PSS) (Cavaliere and Pezzotta 2012; Hein et al. 2018). PSS combine tangible products with intangible services to produce value for the customer. An example is Rolls-Royce's ‘power-by-the-hour’ concept that provides gas turbines (the product) bundled with services including maintenance and improvements so that customers pay for operating hours rather than just for the physical product (Baines et al. 2007). The notion of PSS has gained further interest with the development of smart services enabled by cyber-physical systems, leading to complex software-product-service systems (Mikusz 2014).

Although systems engineering has matured as a proper design discipline with its own specialised tools and approaches (MIT 2003), its grounding in design theory has been insufficiently established. In particular, there is a scarcity of research examining the commonalities and differences between the systems engineering process and models of designing in other disciplines. Existing studies focus on the unique skills and competences required in systems

engineering, especially with respect to those in design thinking (Greene et al. 2019; Li 2002; Patel and Mehta 2017; Pourdehnad et al. 2011). Yet, there has not been any detailed comparison of the process models used in the different domains, except for a few hypothesised mappings between process phases across different models on a high level. For example, Chang et al. (2008) present a simplified model of systems engineering consisting of iterative “functional analysis”, “synthesis” and “evaluation and decision” phases that correspond to Asimov's (1962) phases of designing.

The aim of this paper is to investigate how a process model of systems engineering is different from other design domains that cover only partial aspects of what constitutes typical systems: mechanical engineering, software engineering and service design. In particular, the core technical processes of systems engineering described in the *Systems Engineering Handbook* published by INCOSE (2015) are compared with Pahl and Beitz' (2007) *Systematic Approach* in mechanical engineering, the Rational Unified Process (RUP) in software engineering (Kruchten 2004), and Design for Six Sigma (DFSS) in service design (El-Haik and Roy 2005). This comparison is part of a larger research project that investigates systems engineering from theoretical and empirical perspectives, to provide a better understanding of systems engineering as a basis for the development of improved tools and education. The model described by INCOSE has been chosen because it is one of the most comprehensive, detailed and widely known models of systems engineering.

In this study, the function-behaviour-structure (FBS) schema (Gero 1990; Gero and Kannengiesser 2004, 2014) serves as the uniform representation needed for comparing the four models of the design process. This approach is similar to the idea of using a common ontological representation for knowledge sharing (Gruber 1995), referring to the meaning of ontology in computer science rather than philosophy. Hence, we will refer to the FBS schema as the FBS ontology. The FBS ontology provides a representation that is not only more abstract but also more fine-grained than the four domain-specific models examined (Wynn and Clarkson 2018). A similar approach is the theory-based analysis proposed by Reich et al. (2012) who use C-K theory as a common representation for comparing different design methods. In our approach, once the models of designing are brought into a common format, statistical analyses are used as a basis for data-driven comparisons. Specifically, the distribution and evolution of FBS design issues across the different process models is examined using a set of statistical analyses, including correspondence analysis, cumulative occurrence analysis and Markov model analysis. The FBS ontology was already used by Kannengiesser and Gero (2015) for comparing Pahl/Beitz, RUP and DFSS-ICOV models, and is a well-established coding schema for analysing empirical data.

Fig. 1 The INCOSE model of systems engineering and other models of designing are brought into a common representation that is used as input for statistical analyses and inter-model comparisons

The basic approach of this paper is shown conceptually in Fig. 1.

The paper has the following structure: Sect. 2 presents the INCOSE model of systems engineering. Section 3 is an initial approach to comparing INCOSE with the other models of designing based on hypothesised mappings between process phases. Section 4 introduces the FBS ontological schema, explains the coding of the models using that schema, and presents the statistical analyses applied to the coded models. Section 5 includes the results of the analyses, which are then discussed in Sect. 6. Section 7 concludes the paper with a summary of the contributions and an outlook on future research.

2 The INCOSE model of systems engineering

2.1 Overview

One of the most detailed models of systems engineering is described in the Systems Engineering Handbook published by the International Council on Systems Engineering (INCOSE 2015). It covers a broad range of system life cycle processes including technical processes, technical management processes, agreement processes, and organizational project-enabling processes. In this paper, we will refer to this model as the INCOSE model. It is the result of synthesising and detailing a number of international systems engineering

standards including ISO/IEC 15288, ISO/IEC/IEEE 29148 and ISO/IEC/IEEE 42010:2011. Based on its fine-grained level of detail and strong foundations in the existing body of knowledge, we selected the INCOSE model to represent systems engineering in our analysis.

A fundamental idea in INCOSE and various other models of systems engineering (VDI 2004; NASA 2007; ISO/IEC/IEEE 2015, 2018; INCOSE 2015) is the assumption of a “Vee model” (Forsberg and Mooz 1991) shown in Fig. 2. In the Vee model, the system to be designed is first decomposed into subsystems and components, which are then realised and integrated back into subsystems and finally the overall system.

The INCOSE model adopts the generic system life cycle stages defined in ISO/IEC/IEEE (2015). They include Concept, Development, Production, Utilization, Support and Retirement (INCOSE 2015, p. 28). Each of these stages contains one or several Vee models according to particular engineering concerns to be addressed using decomposition and recomposition; for example, the Development stage may include two Vee models: one for developing (de- and recomposing) a system prototype and one for developing (de- and recomposing) a pre-production prototype (ISO/IEC 2002, p. 40).

The INCOSE model defines 14 technical processes representing the core activities in systems engineering, from business analysis and requirements definition to production, usage and disposal:

Fig. 2 The Vee model (based on Forsberg and Mooz (1991))

1. Business or mission analysis
2. Stakeholder needs and requirements definition
3. System requirements definition
4. Architecture definition
5. Design definition
6. System analysis
7. Implementation
8. Integration
9. Verification
10. Transition
11. Validation
12. Operation
13. Maintenance
14. Disposal

2.2 Scoping and assumptions

For the purposes of this paper, only a subset of the technical processes is considered: those having a similar scope as the models of designing in other domains. In such models, the elicitation of requirements is typically not part of the scope, as designing is viewed as commencing with an initial set of given requirements. Therefore, for reasons of comparability across different models, the two elicitation processes defined in INCOSE—business or mission analysis and stakeholder needs and requirements definition—are not within the scope of our analysis. The first INCOSE process taken into account is system requirements definition, which aims to “transform the stakeholder, user-oriented view of desired capabilities into a technical view of a solution that meets the operational needs of the user” (INCOSE 2015, p. 57). Restricting our scope of analysis in this way is also based on the larger project in which this study is embedded, where the aim is to

compare the INCOSE model with experimental studies of systems engineers. In these studies, the designers are already provided with a set of requirements.

The scope of analysis also needs to be restricted at the other end of the systems engineering process. Other models of designing as well as experiments commonly terminate when a description of a design structure has been produced, assessed and finalised. The subsequent use of that description for manufacturing, physical installation, operation, etc., is not considered. Accordingly, the INCOSE processes of transition, operation, maintenance and disposal are disregarded.

Another INCOSE process, namely system analysis, has been omitted in our study. It is described in INCOSE (2015, p. 74) as a supporting process that can be used by several of the other technical processes, including system requirements definition, architecture definition, design definition, integration, verification and validation. Yet, in the detailed description of these processes no exact location is provided where the system analysis process shall be invoked. In addition, many of the steps defined in system analysis resemble or are identical with the steps already defined in the other processes.

Given these scope restrictions, the INCOSE technical processes taken into account include:

1. System requirements definition (shorthand: Sys Req Def)
2. Architecture definition (shorthand: Arch Def)
3. Design definition (shorthand: Design Def)
4. Implementation (shorthand: Impl)
5. Integration
6. Verification

7. Validation

For reasons of presentational simplicity, we will keep referring to this subset as the “INCOSE model”, being aware a few of its technical processes are omitted in our study.

Processes 1–3 are located at the downward side of the Vee, process 4 at the base, and processes 5–7 at the upward side. According to INCOSE (2015), they are executed in sequence but using iteration and recursion. Iteration is defined as “the repeated application of and interaction between two or more processes at a given level in the system structure or hierarchy” and recursion as “the repeated application of and interaction of processes at successive levels in the system structure” (INCOSE 2015, p. 32). Based on ISO/IEC (2002), recursion occurs when going downward and upward along the Vee. The base of the Vee represents the lowest level in the system hierarchy and therefore does not include any recursion. The overall sequence of INCOSE processes including the resulting recursions (which are a function of the number of system levels) is shown in Fig. 3.

The exact sequence of technical processes within an iteration is not clearly specified in the INCOSE model. The iterations shown in Fig. 3 are simplified to represent a circular sequence, for the purpose of locating them in the Vee model and distinguishing them from recursion. However, the INCOSE model suggests more complex iterations based on various interactions between the processes involved (INCOSE 2015, p. 33). We explore two variants of iteration shown conceptually in Fig. 4 for three generic processes A, B and C. These generic processes are placeholders for either System Requirements Definition, Architecture Definition and Design Definition (i.e., iteration at the downward side of the Vee), or for Integration, Verification and Validation (i.e., iteration at the upward side of the Vee).

Fig. 4 Two generic patterns of iteration explored in this study: **a** cascading, and **b** figure-of-eight

3 Qualitative comparison with models in other domains

In this section, a first attempt is made to relate the INCOSE model with models of designing in other domains, based on a qualitative comparison. Such a comparison has already been carried out by Kannengiesser and Gero (2015) for Pahl and Beitz’ (2007) Systematic Approach, the Rational Unified Process (RUP) (Kruchten 2004), and the Identify-Conceptualize-Optimize-Validate variant of Design for Six Sigma (DFSS/ICOV) (El-Haik and Roy 2005). We will first review the similarities among these models before relating the INCOSE model to them.

As shown by Kannengiesser and Gero (2015), all three models suggest a four-phase process. While their domains and terminologies are different, the phases can clearly be mapped onto one another as they share common overall

Fig. 3 Iteration and recursion in the INCOSE model: Iterations occur within every system hierarchy level (1 to N) along the downward and upward sides of the Vee. Recursions (R_1 to R_{N-1}) occur when transitioning across adjacent system hierarchy levels via repeated process cycles

goals. The goal of the first phase can be viewed as understanding and defining the design problem. This is done in all three models by collecting, analysing and documenting the requirements and scope of the design. The second phase is concerned with generating a conceptual structure. This is a common goal of all three models, despite their focus on different kinds of structure according to their specific domains (i.e., mechanical structure, software architecture and service structure). The same holds true for the third phase: It generates a concrete solution structure, using the different types of components and materials of the specific design domains (e.g., physical layout, software components and service configurations). The fourth phase aims to finalise and deliver the design solution. The particular emphasis of this phase may vary among the models, but they all include some form of refinements, quality checks and documentation.

Some of the technical processes—from now on called “phases”—in the INCOSE model appear to naturally fit with these mappings. Specifically, it is hypothesized that System Requirements Definition, Architecture Definition, Design Definition and Implementation can be mapped onto the four design phases in the other models, as shown in Table 1.

Three of the seven phases of the INCOSE model—Integration, Verification and Validation—do not fit into this four-phase process structure. This indicates that, while there is some similarity between INCOSE and the other models, there is also some difference. This qualitative assessment will be examined quantitatively in the remainder of this paper.

4 Methodology

The methodology for this research follows the approach presented in Fig. 1, consisting of two phases: (1) bringing the different models of designing into a single, uniform representation based on the FBS ontology, and (2) running a set of statistical analyses on these representations. This section describes the two phases in more detail.

4.1 Uniform representation based on the FBS ontology

For the transformation of domain-specific models of designing into a canonical format, two mappings are needed (Kannengiesser and Gero 2015). Firstly, a mapping of the specific concepts and terms used in the different models onto FBS design issues, and secondly, a mapping of the design steps described in each model onto the situated FBS framework as a basis for a simulation model.

The FBS design issue schema allows generalising the specifics of domains into a generic representation in terms of six classes of concepts: requirements (R), function (F), expected behaviour (Be), behaviour derived from structure (Bs) (or, shorthand, structure behaviour), structure (S), and description (D). Here, we will provide an overview of how the concepts of INCOSE were mapped onto FBS. For the FBS mappings of Pahl/Beitz, RUP and DFSS/ICOV, readers may refer to Kannengiesser and Gero (2015). The terminology used for describing the FBS schema in this paper is taken from Gero and Kannengiesser (2014) and Kannengiesser and Gero (2015).

Requirements (R): include articulated customer or market needs, demands, wishes and constraints, explicitly provided at the outset of a design task. In INCOSE, requirement issues comprise concepts such as “stakeholder requirements”, “system requirements”, and “interactions of the system with systems external to the system boundary” and other “constraints” stated as input to the design process (INCOSE 2015, p. 59).

Functions (F): include articulations of intent related to the artefact. They are not externally provided (otherwise they would be requirements) but are generated by the designer. Function issues in INCOSE comprise “system functions”, “critical quality characteristics” (e.g., “safety, security, reliability, supportability”) and other teleological aspects of a system (*ibid*, p. 59).

Expected behaviour (Be): includes attributes describing the artefact’s expected interaction with the environment, providing measurable assessment criteria for design candidates.

Table 1 Hypothesised mappings between the design phases in INCOSE, Pahl/Beitz, RUP and DFSS/ICOV (based on Kannengiesser and Gero (2015))

Mapping No	INCOSE	Pahl/Beitz	RUP	DFSS/ICOV	Overall goal
1	Sys Req Def	Task Clarification	Inception	Identify	Understanding and defining the design problem
2	Arch Def	Conceptual Design	Elaboration	Conceptualize	Generating a concept structure
3	Des Def	Embodiment Design	Construction	Optimize	Generating a solution structure
4	Impl	Detail Design	Transition	Validate	Finalising and delivering the design solution

Expected behaviour issues in INCOSE comprise “operational scenarios and expected system behaviours” (*ibid*, p. 66), “architecture evaluation criteria” (*ibid*, p. 67) and “performance characteristics” (*ibid*, p. 84).

Structure behaviour (Bs) (or “Behaviour derived from Structure”): includes attributes that are measured, calculated or derived from observation of a specific design solution and its interaction with the environment. The comparison of structure behaviour and expected behaviour is the basis for evaluating design solutions. Structure behaviour issues cover the same concepts in INCOSE as outlined for expected behaviour issues.

Structure (S): includes the components of an artefact and their relationships. Structure can exist on a conceptual and a more detailed level. In INCOSE, structure issues comprise concepts such as “system elements”, “physical interfaces” and other “architectural entities” (*ibid*, p. 66) that are defined or implemented.

Description (D): includes any form of external design representations produced during the design process. Description issues in INCOSE comprise all forms of “outputs” defined for the different technical processes. For example, the outputs of Design Definition include “system design description”, “system design rationale”, “interface definition” and others (*ibid*, p. 71).

Bringing the different models of designing into a common, procedural representation requires mapping the steps specified in each model onto the 20 processes defined in the situated FBS (sFBS) framework (Kannengiesser and Gero 2015). This allows viewing them as part of the interaction between a design agent and the design situation, resulting in simulation models of the respective design processes. The design situation is defined as the interaction between three “worlds”: 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. The sFBS framework and rules for mapping the 20 sFBS processes onto the six basic FBS design issues are shown in Fig. 5 and Table 2, respectively.

The approach can be illustrated using the initial design activities in the INCOSE System Requirements Definition phase, shown in Table 3. The first activity within this phase is to “prepare for system requirements definition” (INCOSE 2015, p. 59). One of the inputs defined for System Requirements Definition are “stakeholder requirements” (*ibid*, p. 58) related to functions, constraints and interactions (*ibid*, p. 54). As they can be assumed to be provided externally to the systems engineer, they are interpreted as requirements (R) via processes 1 and 2 in the sFBS framework. The requirements are then complemented by determining “the system boundary, including the interfaces, that reflects

the operational scenarios and expected system behaviors” (INCOSE 2015, p. 59), which can be mapped onto the construction of expected behaviour (Be) (process 5 in sFBS). This activity also includes the identification of “expected interactions of the system with systems external to the system (control) boundary as defined in negotiated interface control documents (ICDs)” (*ibid*, p. 59) that are part of the external requirements (R) on behaviour (process 2 in sFBS). The following INCOSE activity is to “define system requirements”, which begins with “identify[ing] and defin[ing] the required system functions” (*ibid*, p. 59). This is mapped onto the interpretation of requirements (R) related to function (process 1 in sFBS) and the construction of further functions (F) (process 4 in sFBS). In addition, “system behavior characteristics” (*ibid*, p. 59), i.e., expected behaviours (Be), are constructed (process 5 in sFBS). Finally, after a few steps similar to the ones already described (and thus omitted in Table 3), system requirements are specified that include “stakeholder requirements, functional boundaries, functions, constraints, critical performance measures, critical quality characteristics” (*ibid*, p. 59). They are defined as the output of the *System Requirements Definition* phase (*ibid*, p. 58) and therefore viewed as external descriptions (D) of function and behaviour (processes 18 and 17 in sFBS).

The complete set of mappings of all INCOSE phases onto sFBS and FBS design issues are depicted in the Appendix. For the mappings of Pahl/Beitz, RUP and DFSS/ICOV onto FBS, readers are referred to Kannengiesser and Gero (2015). All FBS coding was carried out by the two authors of this study, who are experts in the FBS ontology. The FBS ontology-based coding scheme has been used by multiple researchers across multiple domains (Bott et al. 2019; Cascini et al. 2012; Pauwels et al. 2015; Song 2014; Hamraz and Clarkson 2015; SAE 1999).

For completing the simulation models, three assumptions are made for all four models of designing: (1) every elementary design step occurs only once within the same iteration (as we abstract from the instance level to a generic model level), (2) both generic patterns (i.e., “cascading” and “figure-of-eight”) described in Sect. 2 are explored as two separate variants, and (3) the number of iterations is set to one (i.e., a single execution of a generic pattern). An additional assumption applies exclusively for INCOSE, setting the number of hierarchy levels (N) to 3, leading to two (N–1) recursions (cf. Figure 3). Setting the number of hierarchy levels to 3 in the INCOSE modelling matches the coding of the empirical data. The other three models do not need such an assumption as they do not have the notion of hierarchy levels.

Based on the mappings and assumptions detailed above, the four models of designing are brought into uniform representations with the following number of steps: INCOSE: 1573, Pahl/Beitz: 185, RUP: 224, DFSS/ICOV: 77. An

Fig. 5 The situated FBS framework (Gero and Kannengiesser 2004, 2014)

overview of how the numbers of steps were calculated for INCOSE is provided in Table 4. The numbers for Pahl/Beitz, RUP and DFSS/ICOV are taken from Kannengiesser and Gero (2015).

4.2 Statistical analyses

Three types of statistical analysis can now be carried out on the common representations: correspondence analysis, cumulative occurrence analysis and Markov model analysis.

Correspondence analysis (CA) is a technique for reducing the dimensionality of categorical data and visualising the results on a two-dimensional plot (Benzecri 2019; Greenacre 2017). The axes of the plot do not have a meaning other than capturing the highest variance of the data. CA is conceptually similar to principal components analysis applied

to categories. In this research, the two types of categories analysed are the set of models of designing (and the phases within these models) and the set of design issues. The CA plot then allows visually interpreting the relationships between the different models and between the models and the design issues.

Cumulative occurrence analysis aggregates the occurrence of a design issue overall design steps in a model to derive a set of measures characterising the resulting curve (shown conceptually in Fig. 6). The following measures have been used in previous work (Kannengiesser and Gero 2015):

- Regression line: includes first-order (linear) and second-order polynomials, where the latter are characterised as either concave (i.e., the graph is flattening over time) or convex (i.e., the graph is rising over time). This allows

Table 2 Mapping of sFBS processes to 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 or Bs (*)
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

making statements about how the focus on each design issue changes over the course of designing.

- Slope: represents the rate at which design issues are generated, using the assumption that the graph is linear.
- First occurrence at start: indicates whether design issues first occur near the start of designing or at a later stage. This is based on past observations in other models of designing that some design issues do not start occurring until later in the design process (Kannengiesser and Gero 2015).

Markov model analysis is based on the probabilities of moving from one state to another (Meyn and Tweedie 2009). Applied to models of designing, it captures the transition probabilities between FBS design issues (Kan and Gero 2009). It increases understanding of the interrelationships of the six design issues in the different models of designing. In this research, we use first-order Markov models that capture the transition probability to a future design issue depending only on the current design issue without considering any past design issues. The models can be produced using the LINKODER tool that outputs a probability matrix for the six design issues.

5 Results

5.1 Correspondence analysis

Correspondence analyses (CA) were carried out based on the numbers of design issues (here called frequency distributions) in different design models and design phases. For these analyses, no iterations in the models were taken into account. This is because frequency distributions represent the total numbers of design issues per model or design phase, and are independent of variations in their accumulation over the course of designing.

The results of the CA for the four models from a global perspective are shown in Fig. 7. Firstly, we look only at the location of the four models (in blue colour) on the biplot. Each of the models is positioned in a separate quadrant, except for Pahl/Beitz and DFSS/ICOV that are at almost identical locations within the same quadrant. On the vertical axis (Dim2), INCOSE sits in the middle between RUP and Pahl/Beitz/DFSS/ICOV. However, on the horizontal axis (Dim1), which accounts for 85.6% of the variance in the data, INCOSE sits almost at the opposite end from RUP, Pahl/Beitz and DFSS/ICOV. This indicates that INCOSE is quite different from the other models from a categorical perspective.

Table 3 Example of the mappings of the INCOSE system requirements definition phase onto the sFBS framework and the FBS design issue system

Step	INCOSE activity	Process in sFBS (label)	FBS issue
1	Prepare for system requirements definition	Interpret requirements on function (1) and behaviour (2)	R
2		Construct expected behaviours not explicitly stated (5)	Be
3		Interpret requirements on behaviour (2)	R
4	Define system requirements	Interpret requirements on function (1)	R
5		Construct functions not explicitly stated (4)	F
6		Construct expected behaviours not explicitly stated (5)	Be
...		...	...
13		Produce external representations of function (18) and behaviour (17)	D
...	...	...	...

Table 4 Calculation of the total number of steps of the coded INCOSE model behaviour (Bs)

Phase	Steps per phase	Recursions	Iterative repetitions per recursion	Total
Sys Req Def	28	2x	2x	112
Arch Def	88	2x	3x	528
Design Def	44	2x	2x	176
Implementation	33	1x	1x	33
Integration	36	2x	2x	144
Verification	60	2x	3x	360
Validation	55	2x	2x	220
Grand total	344			1573

Fig. 6 Conceptual graph showing the cumulative occurrence of a design issue

Secondly, we look at the relationships between the four models of designing (blue colour) and the six design issues (red colour). We do this by imagining lines connecting the models or design issues with the origin of the biplot. The longer the line connecting a model and the line connecting a design issue, and the smaller the angle between the two lines, the stronger the association between the model and the design issue (cf. Greenacre (2017)). Thus, from Fig. 7 we can infer a moderate to strong association between INCOSE and requirements (R) and structure behaviour (Bs). In contrast, RUP is negatively associated with R, and Pahl/Beitz and DFSS/ICOV are negatively associated with Bs, based on their location at opposite sides of the origin (i.e., angle of 180°). These negative associations are quite significant, as the respective locations on the biplot are a long way from the origin. In other words, INCOSE makes use of requirements (R) and structure behaviour (Bs) much more than the other models do.

The results of CA for the individual phases in INCOSE and those in Pahl/Beitz, RUP and DFSS/ICOV are shown in Fig. 8. The phases of INCOSE that clearly stand out are Verification and Validation, as they are the only ones located in the fourth quadrant apart from RUP's Transition. In addition,

there is a strong association between Validation and structure behaviour (Bs), and, slightly less, between Verification and Bs. No other phases in any of the models exhibit these associations with Bs. System Requirements Definition and Architecture Definition sit in the first quadrant, and so do Task Clarification (Pahl/Beitz), Inception (RUP) and Identify (DFSS/ICOV). Design Definition, Implementation and Integration are located in the second quadrant, which they share only with Conceptual Design of Pahl/Beitz. The third quadrant is where all the remaining phases of Pahl/Beitz, RUP and DFSS/ICOV are located, and none of INCOSE. These results provide only partial confirmation of the mappings identified qualitatively between the phases across the different models (see Table 1). However, the overall sequence of the phases in each model is roughly mirrored by their clockwise positioning from the first to the fourth (for INCOSE and RUP) or third (for the other models) quadrants. A similar pattern can be discerned from the clockwise positioning of the six design issues that roughly follows their expected ordering according to the FBS framework: $R \rightarrow F \rightarrow Be \rightarrow S \rightarrow Bs \rightarrow D$ (with the exception that in the plot D comes before Bs).

5.2 Cumulative occurrence analysis

In this section the cumulative occurrence graphs of the four models (INCOSE, Pahl/Beitz, RUP and DFSS/ICOV) are presented and compared. The graphs for cascading and figure-of-eight iterations are shown in Figs. 9 and 10, respectively, for readers to appreciate the variability of the data.

Regression lines were calculated for the cumulative occurrence graphs using standard spreadsheet software. The shapes of the graphs are characterised in Tables 5 and 6 for the two iteration types as either linear (i.e., constant), concave (i.e., flattening) or convex (i.e., rising). They can be used as a measure for the similarities and differences between INCOSE and the other models, independently of the type of iteration used. The similarities include linear requirements (R) and structure (S) (except for DFSS/ICOV (cascading iterations) that has a convex curve), and concave function (F) and expected behaviour (Be) (except for RUP that has linear F and Be (figure-of-eight iterations)). INCOSE differs from the other models in the convexity of structure behaviour (Bs) and description (D) (except for RUP that has a linear curve for Bs (figure-of-eight iterations)). To summarise, as the design process proceeds, INCOSE generates more structure behaviour (Bs) and description (D) issues at the expense of function (F) and expected behaviour (Be) issues. The other models absorb the decreasing production of F and Be issues more equally within the remaining design issues, leading to no significant increase in Bs and D issues.

Fig. 7 Correspondence analysis of the models and design issues

The convexity of structure behaviour (Bs) and description (D) in INCOSE emerges from a change in the shape of the corresponding graphs occurring after a little more than half of the design steps. This is where the INCOSE process enters the final phases of Integration, Verification and Validation (from step 850). If these phases were not considered in calculating the regression lines, Bs and D would be linear just like in the other models of designing.

The slopes of the cumulative occurrences can be interpreted as showing their relative importance within a model. They are depicted graphically in Fig. 11 for the two iteration types, using the simplifying assumption that all regression lines are linear. With respect to the other models, in INCOSE the slopes are higher for requirements (R) and structure behavior (Bs), while they are slightly lower for function (F) and description (D). The slope is also lower for structure (S) in case of cascading iterations. Overall, it can therefore be stated that INCOSE emphasizes R and Bs issues more than the other models do.

The first occurrence of design issues near the start of the models is shown in Table 7. It is based on whether the design issue occurs in the first phase of the model, which is independent of the iteration type. In all four models, structure behavior (Bs) and structure (S) do not occur near the start, whereas the other design issues do.

5.3 Markov model analysis

First-order Markov models have been established for INCOSE, Pahl/Beitz, RUP and DFSS/ICOV (covering cascading and figure-of-eight iterations). The dominant state transitions in every model—i.e., those transitions with the highest likelihood of all other transitions from a given design issue—are represented graphically in Fig. 12.

A number of similarities and differences between INCOSE and the other models can be identified by comparing their dominant state transitions, as summarised in Table 8. INCOSE is similar to all other models in the

Fig. 8 Correspondence analysis of all phases in INCOSE, Pahl/Beitz (shorthand: PB), RUP and DFSS/ICOV (shorthand: DFSS)

transitions from requirements (R) (to function (F)), structure behaviour (Bs) (to structure (S)) and description (D) (to structure (S)). It is partially similar to the other models in that it shares with Pahl/Beitz and DFSS/ICOV the dominant transition from function (F) to expected behaviour (Be), and with Pahl/Beitz and RUP the transition from expected behaviour (Be) to structure (S). It differs from all other models in its dominant transition from structure (S) to structure.

6 Discussion

The results of the statistical analyses show that there are both similarities and differences between INCOSE and the other models of designing.

When comparing the overall models and their phases by reducing the dimensionality of the data (using CA), only a small number of similarities can be observed. One similarity

is that the first phase of every model is located in the same quadrant of the CA plot. This confirms the hypothesised mapping between these phases laid out in Table 1. However, the other mappings from that Table are not confirmed by the CA.

One of the similarities identified through cumulative occurrence analysis is that structure (S) is generated at a constant rate in both INCOSE and the other models (with the exception of one configuration of the DFSS/ICOV model). Problem-related issues (i.e., F and Be) are generated more at the beginning and less towards the end in INCOSE and most of the models (except in RUP where they are constantly generated). The solution-related issues of S and Bs do not occur at the beginning of any of the models. Taken together, these similarities indicate that INCOSE follows the general “problem to solution” pattern in design, characterised by a shift in focus from problem-oriented (F, Be) towards solution-oriented issues (Bs) during the course of design.

Fig. 9 Cumulative occurrence of design issues for cascading iterations

Fig. 10 Cumulative occurrence of design issues for figure-of-eight iterations

In terms of the transition probabilities of individual design issues, INCOSE shares with the other models that the dominant transition from R is to F, from Bs is to S and from D is to S. It shares with two of the other models that the dominant transition from F is to Be and from Be is to S. This means that five out of the six FBS design issues in INCOSE have the same dominant transitions as in (most of) the other models of designing, which indicates quite a strong similarity.

On the other hand, the statistical analyses reveal a number of distinct characteristics of INCOSE that cannot be found in the other models.

One result of CA is that the Verification and Validation phases are clearly set apart from all others and strongly associated with structure behaviour (Bs). The CA of the overall models of designing also shows a moderate to strong connection between INCOSE and Bs issues in addition to R issues, which supports the conclusions drawn from the comparison of slopes of the cumulative occurrence graphs. Overall, the results indicate a fundamental difference between INCOSE and the other models, as they are located at different ends of the major dimension (Dim1) in the CA plot.

The accumulation of structure behaviour (Bs) and design description (D) issues in INCOSE is convex, i.e., increasing over time. This is different from the other models where these issues are generated linearly (with the exception of one configuration of RUP where the Bs issues also increase). The cumulative occurrence of R issues in INCOSE is linear,

which seems unique when visually inspecting the other graphs; however, it should be noted that those other graphs lack sufficient data to support this statement. The comparison of the slopes of the different graphs has shown that the importance of R and Bs issues is slightly emphasised in INCOSE.

Despite the strong similarity related to transition probabilities, INCOSE differs in one aspect: It is the only model that has a dominant state transition from structure (S) to structure behaviour (Bs). A look at the raw data (i.e., the coded INCOSE model, see Appendix) shows that this transition occurs most often within the phases of Verification and Validation. It confirms the findings from CA that these phases are quite different from other phases within INCOSE and from phases within other models of designing.

Finally, two limitations of this study should be stated: firstly, the INCOSE model was reduced to those technical processes transforming stakeholder requirements into documents of the system designed, to align its scope with other models and most empirical design sessions (see Sect. 2.2). The INCOSE processes of business or mission analysis and stakeholder needs and requirements definition were thus ignored. Being mainly concerned with the elicitation of requirements, their inclusion in the statistical analysis would probably have led to similar results in terms of the INCOSE's strong emphasis on R issues. Secondly, this study explored only two types of iteration of the individual processes within INCOSE. Yet, the results are rather insensitive

Table 5 Regression lines of the cumulative occurrence graphs (cascading iterations)

	R	F	Be	Bs	S	D
INCOSE	Linear	Concave	Concave	Convex	Linear	Convex
Pahl/Beitz	Linear*	Concave	Concave	Linear	Linear	Linear
RUP	—**	Concave	Linear	Convex	Linear	Linear
DFSS/ICOV	—**	Concave	Concave	—**	Convex	Linear

*Calculated based on only nine occurrences

**Not calculated due to less than nine occurrences

Table 6 Regression lines of the cumulative occurrence graphs (figure-of-eight iterations)

	R	F	Be	Bs	S	D
INCOSE	Linear	Concave	Concave	Convex	Linear	Convex
Pahl/Beitz	Linear*	Concave	Concave	Linear	Linear	Linear
RUP	—**	Linear	Linear	Linear	Linear	Linear
DFSS/ICOV	—**	Concave	Concave	—**	Linear	Linear

*Calculated based on only nine occurrences

**Not calculated due to less than nine occurrences

Fig. 11 Graphical comparison of slopes of the cumulative occurrence graphs

Table 7 First occurrence at start

	R	F	Be	Bs	S	D
INCOSE	Yes	Yes	Yes	No	No	Yes
Pahl/Beitz	Yes	Yes	Yes	No	No	Yes
RUP	Yes	Yes	Yes	No	No	Yes
DFSS/ICOV	Yes	Yes	Yes	No	No	Yes

as to whether “cascading” or “figure-of-eight” iterations were used.

7 Conclusion

This paper investigated the question whether a model of systems engineering (INCOSE) is different to models of

Fig. 12 Graphical representation of the dominant state transitions in the Markov models of INCOSE, Pahl/Beitz, RUP and DFSS/ICOV

Table 8 Summary of the similarities and differences between INCOSE and the other models with respect to the dominant state transitions

	INCOSE	Pahl/Beitz	RUP	DFSS/ICOV
From:	To:			
R	F	F	F	F
F	Be	Be	F	Be
Be	S	S	S	F, D
Bs	S	S	S	S
S	Bs	D	D	S
D	S	S	S	S

Deviations of the other models with respect to INCOSE are highlighted

engineering design (Pahl/Beitz), software design (RUP) and service design (DFSS/ICOV). It revealed both similarities

and differences between them, not based on qualitative judgements but on quantitative analyses using statistical methods. The approach is possible because it increases both the level of detail and the level of abstraction: the models of designing are broken down into sequences of fine-grained steps according to the sFBS framework and coded uniformly using the FBS design issue schema. The resulting sets of code are used as datasets on which multiple statistical analyses are run, allowing for data-driven formation and testing of hypotheses about the models. The same approach was used by Kannengiesser and Gero (2015) but has not been applied to the INCOSE model and used only cumulative occurrence analysis.

The combination of multiple types of statistical analysis provides detailed insights in the INCOSE model. The main similarities with other models of designing that were identified include:

- the initial phase of understanding and defining the design problem, whose design steps are quite distinct from other phases,
- the shift in focus during the design process from the design problem to possible solutions,
- the constant generation of solution structures (S), and
- the same dominant state transitions for the majority of design issues.

However, INCOSE also has a set of unique features that make it fundamentally different from the other models. They include:

- a stronger association of the overall model with requirements (R) and structure behaviour (Bs),
- Verification and Validation phases whose design steps are fundamentally different from all others,
- the increased generation of structure behaviour (Bs) and design descriptions (D) towards the end of the design process, and
- the dominant state transition from structure (S) to structure behaviour (Bs).

Requirements (R) keep occurring in most INCOSE phases, whereas in the other models of designing they occur much less frequently and almost only in the initial phases. A large number of structure behaviours (Bs) are produced in INCOSE's Verification and Validation—first-class phases of design analysis that Pahl/Beitz and RUP subsume as sub-activities. DFSS/ICOV does have an explicit validation phase but is not very detailed compared to INCOSE.

Many of the similarities and differences are not surprising with respect to commonly known qualitative views of systems engineering, including its strong focus on requirements definition, on verification and validation, and on documentation. These activities are grounded in the inherent complexity of engineered systems and in the origin of systems engineering in safety-critical domains. However, the analysis in this paper reveals insights beyond these general views. For example, the hypothesized mappings between the phases across the models (see Table 1) were only partially supported by the data. This does not mean that they are not associated by common goals—which can only be interpreted qualitatively, not statistically. The mappings may still be valid teleologically, but they are not based on structural similarity of the phases mapped. Other characteristics of INCOSE have been identified that cannot be derived from current literature on systems engineering. In particular, the constant generation of solution structures during the INCOSE process has not previously been stated. Further studies may shed more light on this phenomenon, which has been observed in experiments and models of designing

across several domains (Gero et al. 2014; Kannengiesser and Gero 2015).

The insights gained in this paper can lay the foundations for better understanding of systems engineering and the INCOSE model, especially by positioning it in the landscape of other models from various design disciplines that may have influenced the development of INCOSE. This may contribute to efforts in building a theory of systems engineering. In addition to principles and hypotheses derived from sources within the systems engineering discipline (Watson 2019), such a theory would incorporate insights based on an external, comparative view. Comparisons may include not only conceptual models from the literature, but also empirical studies. The approach presented in this paper can accommodate both, as was shown by Kannengiesser and Gero (2017) who compared Pahl/Beitz' model with design sessions involving mechanical engineering students. A comparison of the INCOSE model with empirical data from systems engineering design sessions is currently under way.

Knowing where the INCOSE model differs from other models of designing can guide future developments in systems engineering methods. The abstract representation of differences in terms of FBS facilitates the retrieval and transfer of methods from other domains. For example, service walkthroughs to increase understanding of user experience when interacting with a system (Blomkvist 2016) may be a useful addition to common validation techniques in systems engineering. They may be found based on their support for S-to-Bs transitions that have been found important in INCOSE. Product-service system design, which brings together two disparate design disciplines, can be analysed using this INCOSE model. PSS design activity can be compared with that for product design to determine whether it is different and, if so, whether different design support tools need to be developed for it.

Appendix A

Coding of the INCOSE model

See Tables 9, 10, 11, 12, 13, 14, 15.

Appendix B

Contingency tables used as input for correspondence analysis

See Tables 16 and 17.

Table 9 System requirements definition (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
1.1 Prepare for system requirements definition	R	1, 2	-Requirements are used as input (see Fig. 1); they include functions, interactions and constraints (p. 54)
	N/A	N/A	-establish the approach used for defining requirements (p.59)
	Be	5	-determine “the system boundary, including the interfaces, that reflects the operational scenarios and expected system behaviors” (p. 59); also includes expected interactions of the system with systems external to the system boundary (as defined in interface control documents (ICDs))
	R	2	
1.2 Define system requirements	R, F	1, 4	-identify and define required system functions; also, include the system behavior characteristics (p. 59)
	Be	5	
	R, F, Be, F, Be	1, 2, 4, 5, 7, 8	-identify requirements that impose unavoidable constraints (p. 59)
	F	4	-identify critical quality characteristics (e.g., safety, security, reliability, supportability) (p. 59)
	N/A	N/A	-identify technical risks
	D	18, 17	- “specify system requirements, consistent with stakeholder requirements, functional boundaries, functions, constraints, critical performance measures, critical quality characteristics, and risks” (p. 59); specification as a document (SyRS) (p. 59)
1.3 Analyze system requirements	F, Be, F, Be	20, 19, 4, 5	- “analyze the integrity of the system requirements” (p. 59)
	D	18, 17	- “provide analysis results to applicable stakeholders” (p. 59)
	N/A	N/A	-negotiate modifications to resolve issues (p. 59)
	Be	10	- “define verification criteria – critical performance measures that enable the assessment of technical achievement” (p. 59)
	D	17	-verification criteria are used as output (see Fig. 1)
1.4 Manage system requirements	N/A	N/A	-ensure agreement among key stakeholders, establish and maintain traceability, ... (p. 59)
	R, F, Be, F, Be, F, Be, D	1, 20, 2, 19, 4, 5, 7, 8, 18, 17	- “Establish and maintain traceability between the system requirements and the relevant elements of the system definition (e.g., stakeholder requirements, [...])” (p. 59); “traceability” is captured in the “updated RVTM” which is documented as an external output (see Fig. 1)

Table 10 Architecture Definition (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
2.1 Prepare for architecture definition	R, F, Be	1, 2, 4, 5	- “Identify and analyze relevant market, industry, stakeholder, organizational, business, operations, mission, legal, and other information that will help to understand the perspectives that will guide the development [...]” (p. 65)
	F, Be	20, 19	- “[...] analyze the system requirements [...] as well as life cycle constraints” (p. 66)
	R, F, Be	1, 2, 7, 8	- “Capture stakeholder concerns related to architecture. Usually, the stakeholder concerns focus on expectations or constraints that span one or more system life cycle stages.” (p. 66); “life cycle constraints” are among the external inputs to the process (see Figs. 1 and 2)
	Be, S	8, 11	- “Establish the approach for defining the architecture. [...] The approach should also include the process requirements (e.g., measurement approach and methods), evaluation (e.g., reviews and criteria) and necessary coordination. Capture the evaluation criteria.” (p. 66) This approach is part of the “architecture definition strategy” (see Fig. 2)= > external output
	Bs	14, 15	
	D	17, 12	
2.2 Develop architecture viewpoints	R, F, Be, S, F, Be, S	1, 2, 4, 5, 6, 7, 8, 9	<p>“Based on the identified stakeholder concerns, establish or identify the associated architecture viewpoints, the supporting kinds of models that facilitate the analysis and understanding of the viewpoint, and relevant architecture frameworks to support the development of the models and views.” (p. 66) “Architecture viewpoints” and “model kinds” are defined in ISO/IEC/IEEE (2011) that is referenced in INCOSE:</p> <p>-Viewpoint= “work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns” (ISO/IEC/IEEE 2011, p. 2)</p> <p>-Model kind= “conventions for a type of modelling; Examples of model kinds include data flow diagrams, class diagrams, Petri nets, balance sheets, organization charts and state transition models” (ISO/IEC/IEEE 2011, p. 2)</p>

Table 10 (continued)

INCOSE activity	FBS code	sFBS step	Comments
2.3 Develop models and views of candidate architectures	R, Be, S	2, 5, 6	- “In conjunction with the system requirements definition process, determine the system context (i.e., how the SOI fits into the external environment) and boundary, including the interfaces, that reflect the operational scenarios and expected system behaviors.” (p. 66)
	F, Be, S	7, 8, 9	- “Determine which architectural entities (e.g., functions, input/output flows, system elements, physical interfaces, architectural characteristics, information/data elements, containers, nodes, links, communication resources, etc.) address the highest priority requirements” (p. 66)
	F, Be, S	7, 8, 9	- “Allocate concepts, properties, characteristics, behaviors, functions, and/or constraints that are significant to architecture decisions of the system to architectural entities.” (p. 66)
	R, F, Be, S, F, Be, S, Be, S, D	1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 18, 17, 12	-Select, adapt, or develop models of the candidate architectures of the system, such as logical and physical models.” “The models to be used are those that best address key stakeholder concerns. Logical models may include functional, behavioral, or temporal models; physical models may include structural blocks, mass, layout, and other physical models” (p. 66)
	F, Be, S, F, Be, S	4, 5, 6, 7, 8, 9	- “Determine need for derived system requirements induced by necessary added architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions)” (p. 66)
	R, F, Bs, S	1, 2, 20, 19, 13	- “Compose views from the models of the architecture candidates. The views are intended to ensure that the stakeholder concerns and critical requirements have been addressed.” (p. 66)
	N/A	N/A	- “For each system element [...] develop requirements corresponding to allocation, alignment, and partitioning of architectural entities and system requirements to system elements.” (p. 66)
	F, Bs, S	4, 5, 6	- “Analyze the architecture models and views for consistency and resolve any issues identified.” (p. 67)
	D, S, Bs, Bs	12, 17, 18, 13, 19, 14, 15	- “Verify and validate the models by execution or simulation [...] and with traceability matrix of OpsCon. Where possible, use design tools to check their feasibility and validity.” (p. 67)

Table 10 (continued)

INCOSE activity	FBS code	sFBS step	Comments
2.4 Relate the architecture to design	S, S	6,9	- “Determine the system elements that reflect the architectural entities.” (p. 67)
	D	18, 17, 12	- “Establish allocation matrices between architectural entities using their relationships.” (p. 67) “Architectural entities” can relate to F, B and S (see 2.3 above); “matrices” are external descriptions
	Be, S, Be, S D	5, 6, 8, 9 17, 12	- “Perform interface definition” (p. 67) “Preliminary interface definition” is part of the output (see Fig. 2)
	S	11	- “Determine the design characteristics that relate to the system elements and their architectural entities” (p. 67)
	F, Be, S, F, Be, S	4, 5, 6, 7 8, 9	- “Determine need for derived system requirements induced by necessary added architectural entities (e.g., functions, interfaces) and by structural dispositions (e.g., constraints, operational conditions)” (p. 67)
	N/A	N/A	- “For each system element [...] develop requirements corresponding to allocation, alignment, and partitioning of architectural entities and system requirements to system elements.” (p. 67)
2.5 Access architecture candidates	Be S, Bs	19 13, 14, 15	- “Using the architecture evaluation criteria, assess the candidate architectures by applying the system analysis, measurement, and risk management processes” (p. 67)
	D	15, 12, 17	- “Select the preferred architecture(s). This is done by applying the decision management process.” (p. 67) The “decision management process” (p. 111/112) includes producing a record of the decision and supporting documentation (p. 112)
2.6 Manage the selected architecture	D	18, 17, 12	- “Capture and maintain the rationale for all selections among alternatives and decisions for the architecture [...]” (p. 67)
	F, Bs, S, F, Be, S	20, 19, 13, 7, 8, 9	- “Manage the maintenance and evolution of the architecture” (p. 67) Involves analyzing impacts of context changes to the architecture, using allocation and traceability matrices (p. 67)
	N/A	N/A	- “Establish a means for the governance of the architecture” (p. 67)
	R, F, Bs, S, F, Be, S	1, 2, 4, 5, 6, 7, 8, 9	- “Coordinate review of the architecture to achieve stakeholder agreement. The stakeholder requirements and system requirements can serve as references.” (p. 67)

Table 11 Design Definition (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
3.1 Prepare for design definition	Be, Be, S	10, 5, 6	- “Plan for technology management. Identify the technologies needed to achieve the design objectives for the system and its system elements.” (p. 72) Examples for “technologies” include mechanics, electronics, software, chemistry, human operations, and services (p. 71)
	S	6	- “Identify the applicable types of design characteristics for each system element [...]” (p. 72)
	Be, S, S, D	8, 9, 11, 12, 17	- “Define and document the design definition strategy, including the need for and requirements of any enabling systems, products, or services.” (p. 72)
3.2 Establish design characteristics and design enablers related to each system element	R, F, Be, Be, S	1, 2, 7, 8, 10, 6	- “Perform requirements allocation to system elements [...]” (p. 72)
	Be, S, D, S, Bs	10, 11, 12, 13, 14, 15	- “Define the design characteristics relating to the architectural characteristics for the architectural entities, and ensure that the design characteristics are feasible. Use design enablers such as models (physical and analytical), design heuristics, etc.” (p. 72) “design characteristics” include S (e.g., shape, p. 73) and B (e.g., resistance to forces, p. 73)
	Be, S, Be, S, D	5, 6, 8, 9, 17, 12	- Perform interface definition; includes both internal and external interfaces (p. 72)
	D	12, 17	- “Capture the design characteristics of each system element.” (p. 72)
	D	12, 17, 18	- “Provide rationale about selection of major implementation options and enablers.” (p. 72)
3.3 Assess alternatives for obtaining system elements	S, Bs	6, 14	- “Identify existing implemented elements [including COTS or reused system elements]. Alternatives for new system elements to be developed may be studied.” (p. 72)
	N/A	15	- “Assess options for the system element [including the COTS elements, reused elements and new elements] using the selection criteria [...]” (p. 72)
	S	11	- “Select the most appropriate alternatives.” (p. 72)
	N/A	N/A	- IF: decision is made to develop the system element CONTINUE; ELSE: Use acquisition process (chp. 6.1) (p. 72) We assume here that the decision is to develop the element

Table 11 (continued)

INCOSE activity	FBS code	sFBS step	Comments
3.4 Manage the design	D	12, 17, 18	-Capture and maintain the rationale for all selections among alternatives [...]” (p. 72)
	F, Bs, S, F, Be, S	20, 19, 13, 7, 8, 9	- “Manage the maintenance and evolution of the design” (p. 72)
	R, F, Be, S, F, Be, S, D	1, 2, 4, 5, 6, 7, 8, 9, 18, 17, 12	- “Establish and maintain bidirectional traceability between the architecture entities [...] to the stakeholder requirements and concerns [...]” (p. 72)
	D	12, 17	- “Provide baseline information for configuration management” (p. 73)
	N/A	N/A	- “Maintain the design baseline and the design definition strategy” (p. 73)

Table 12 Implementation (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
5.1 Prepare for implementation	S, S, Be, Be, D	13, 6, 5, 8, 12, 17	- “Define fabrication/coding procedures, tools and equipment to be used, implementation tolerances, and the means and criteria for auditing configuration of resulting elements to the detailed design documentation.” (p. 78) Input includes the system design description (p. 77)
	R, Be, S, D	2, 3, 8, 9, 17, 12	- “Elicit from stakeholders, developers and teammates any constraints imposed by implementation technology, strategy, or implementation enabling systems. Record the constraints for consideration in the definition of the requirements, architecture, and design” (p. 78)
	Be, S, Be, S, D	5, 6, 8, 9, 17, 12	- “Document the plan for acquiring or gaining access to resources needed during implementation. The planning includes the identification of requirements and interfaces for the enabling system.” (p. 78)
5.2 Perform implementation	Be, Be	5, 8	- “Develop data for training users on correct and safe procedures for operating and maintaining that element [...]” (p. 78)
	D	12, 17	- “Complete detailed product, process, material specifications (“build-to” or “code-to” documents), and corresponding analyses.” (p. 78)
	S, Bs, D	13, 14, 15, 17	- “Ensure the realization of the system elements per the detailed product, process, material specifications and produce documented evidence of implementation compliance [...]” “Conduct peer reviews and testing”, “Conduct hardware conformation audits” (p. 78)
	D	17	- “Prepare initial training capability and draft training documentation [...]” (p. 78)
	D	12	- “Prepare a hazardous materials log, if applicable.” (p. 78)
	Be, Be	5, 8	- “Determine the packaging and storage requirements for the system element [...]” (p. 78)
5.3 Manage results of implementation	D	12, 17	- “Identify and record implementation results.” (p. 78)
	D	15, 17	- “Record any anomalies encountered during the implementation process, and analyze and resolve the anomalies [...] using the quality assurance process.” (p. 78)
	R, Be, S, Be, S, D	2, 3, 5, 6, 8, 9, 17, 12	- “Establish and maintain traceability of the implemented system elements with the system architecture, design, and system and interface requirements that are needed for implementation.” (p. 78)
	D	12, 17	- “Provide baseline information for configuration management.” (p. 78)

Table 13 Integration (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
6.1 Prepare for integration	S, Be, F, S, Be, F	6, 5, 4, 9, 8, 7	- “Define critical checkpoints to provide assurance of the correct behavior and operation of interfaces and functions of the system elements.” (p. 79)
	S, Be, S, Be, D	6, 5, 9, 8, 6, 5, 12, 17	- “Establish the integration strategy that minimizes integration time, costs, and risks.” (p. 79) Integration strategy is among the outputs of the process (p. 80)
	F, Be, S, F, Be, S, D	16, 5, 6, 7, 8, 9, 18, 17, 12	- “Identify integration constraints on the SOI, arising from the integration strategy, to be incorporated in the system requirements, architecture, and design [...]” (p. 80) E.g., accessibility, safety for integrators, required interconnections of set of system elements (p. 80) Integration constraints are among the outputs of the process (p. 80)
	N/A	N/A	- “The acquisition of the enablers can be done through various ways [...]” (p. 80)
6.2 Perform integration	S, Be, S	13, 19, 6	- “Assemble the verified and validated system elements to form the incremental aggregate using the defined assembly procedures, the related integration enabling systems, and the interface control definitions.” (p. 80)
	Bs, F	14, 15, 16	- “Invoke the system V&V processes, as needed, to check the correct implementation of architectural characteristics and design properties and to check that the individual system elements provide the functions intended.” (p. 80)
6.3 Manage results of integration	D, R, Be, S, Be, S, D	12, 17, 2, 3, 5, 6, 8, 9, 17, 12	- “Identify and record the results of integration. Maintain bidirectional traceability of the updated integrated system elements with the updated system architecture, design, and system and interface requirements. Maintain the records, including configuration updates [...]” (p. 80)
	D	15, 17	- “Record anomalies observed during the integration process [...], and resolve them using the quality assurance process [...]” (p. 80)
	S, Be, S, Be, D	6, 5, 9, 8, 12, 17	- “Update the integration strategy and schedule according to the progress of the project; in particular, the order of system elements assembly can be redefined or rescheduled because of unexpected events or unavailability of system elements as planned.” (p. 81)
	N/A	N/A	- “Coordinate integration activities with the project manager [...]” (p. 81)

Table 14 Verification (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
7.1 Prepare for verification	N/A	N/A	- “Develop a strategy that prioritizes the verification actions to minimize costs and risks while maximizing operational coverage of system behaviors.” (p. 83)
	R, S	1, 2, 13	- “Establish a list of the items for verification, including requirements, architectural characteristics, or design properties, and define the corresponding verification actions.” (p. 83/84); “The approach to verification should be identified and documented [...] to ensure that the requirement as written is indeed verifiable. This may require restating a requirement or decomposing it into several verifiable statements” (p. 84)
	Bs, S, Bs, D, Be, D	14, 9, 8, 12, 17, 10, 17	
	R, F, Be, S, F, Be, S, D	2, 4, 5, 6, 7, 8, 9, 18, 17, 12	- “Establish a list of verification constraints that need to be considered” (p. 84); They include “contractual constraints, limitations due to regulatory requirements [...]”; “Verification constraints” are among the outputs of the process (p. 83)
	F, Be, S, Bs	20, 19, 13, 14	- “Considering the constraints, plan for the methods or techniques that will be applied” (p. 84)
	S, Bs	6, 5	- “Establish the scope of the verification. [...] The selection of what must be verified should be made according to the type of system, the objectives of the project, and the acceptable risks [...]” (p. 84) “Scope” is part of the strategy that is an external output of Verification (p. 83)
	S, Bs, D	9, 8, 12, 17	
	S, Bs, S, Bs	6, 5, 9, 8	- “Develop the verification procedures that support the verification actions. Schedule the execution of verification actions in the project steps and define the configuration of submitted items to verification actions.” (p. 84)
	D	12, 17	“Verification procedure” is among the outputs of the process (p. 83)
	Be, S, Be, S, D	5, 6, 8, 9, 17, 12	- “Identify verification constraints on the system or system elements [...]. Typical constraints include performance characteristics, accessibility, and interface characteristics. Provide the constraint information for consideration in the system requirements definition, architecture definition, and design definition process.” (p. 84)
7.2 Perform verification	N/A	N/A	- “Ensure that the necessary enabling systems [...] are available.” (p. 84)
	D	12, 17	- “Implement the verification plan [...]. That plan includes detailed descriptions for the selected verification actions” (p. 84)
	S, Be, Bs, Bs, D	13, 19, 19, 14, 17	- “Using the verification procedures [including the items to be verified, expected results/criteria, verification method], execute the verification actions and record the results.” (p. 84)
	N/A	15	- “Analyze the verification results against any established expectations and success criteria” (p. 84)

Table 14 (continued)

INCOSE activity	FBS code	sFBS step	Comments
7.3 Manage results of verification	R, S, Bs, D	1, 2, 9, 8, 12, 17	- “Identify and record verification results and enter data in the Requirements Verification and Traceability Matrix (RVTM)” (p. 84)
	D, Bs, Bs	17, 19, 5	- “Record anomalies observed during the verification process, and analyze and resolve the anomalies (corrective actions and improvements) [...]” (p. 84)
	R, Bs, S, Bs, S, D	2, 5, 6, 8, 9, 17, 12	- “Establish and maintain bidirectional traceability of the verified system elements with the system architecture, design, and system and interface requirements that are needed for verification.” (p. 84)
	D	12, 17	- “Provide baseline information for configuration management.” (p. 84)
	S, Bs, S, Bs, D	6, 5, 9, 8, 12, 17	- “Update the verification strategy and schedule according to the progress of the project; in particular, planned verification actions can be redefined or rescheduled as necessary.” (p. 85)
	N/A	N/A	- “Coordinate verification activities with the project manager [...]” (p. 85)

Table 15 Validation (numbers refer to sFBS process labels; page numbers refer to INCOSE (2015))

INCOSE activity	FBS code	sFBS step	Comments
8.1 Prepare for validation	N/A	N/A	- “Establish the validation strategy [...] including: -Identify the stakeholders” (p. 90)
	S, Bs, D	9, 8, 12, 17	- “The scope of the validation plan [relating to] the full system, a system element, or an artefact [...] in addition to the delivered system.” (p. 90)
	R, F, Be, S, F, Be, S, D _F , D	2, 4, 5, 6, 7, 8, 9, 18, 17, 12	- “Establish a list of validation constraints that need to be considered. The constraints [...] include contractual constraints, limitations due to regulatory requirements [...]” (p. 90); Validation constraints are among the outputs of the process (p. 90)
	F, Be, S, Bs	20, 19, 13, 14	- “With appropriate consideration to the constraints, select suitable validation approach” (p. 90)
	S, Bs	9, 8	- “It may be necessary to prioritize the validation actions [...] against constraints, risks, type of system, project objectives, and other relevant criteria” (p. 91)
	F	16	- “Determine if there are any validation gaps and that the resulting validation actions will provide an acceptable level of confidence that the system or system element will meet the identified needs.” (p. 91)
	R, F, Bs, S	1, 7, 8, 9	- “Ensure appropriate scheduling” (p. 91)
	N/A	N/A	- “Define the configuration of submitted items to validation actions.” (p. 91)
	R, S, Bs, S, Bs	2, 13, 19, 6, 5	- “Identify validation constraints on the system, arising from the validation strategy, to be incorporated in the stakeholder requirements.” (p. 91)
	N/A	N/A	- “Ensure that the necessary enabling systems [...] are available” (p. 91)
8.2 Perform validation	S, Bs, S, Bs, D	6, 5, 9, 8, 12, 17	- “Develop the validation procedures that support the validation actions.” (p. 91) Validation procedure is among the outputs of the process (p. 90)
	N/A	N/A	- “Ensure readiness to conduct validation” (p. 91)
	S, Bs, Bs, D	13, 19, 14, 17	- “Conduct validation actions in accordance with the procedures. [...] During the conduct of validation actions, record the results [...]” (p. 91)

Table 15 (continued)

INCOSE activity	FBS code	sFBS step	Comments
8.3 Manage results of validation	R, S, Bs, D _S	1, 2, 9, 8, 12, 17	- “Identify and record validation results and enter data in the validation report (including any necessary updates to the RVTM)” (p. 91)
	D, Bs, Bs	17, 19, 5	- “Record anomalies observed during the validation process, and analyze and resolve the anomalies (corrective actions and improvements) [...]” (p. 91)
	N/A	15	- “Compare the obtained results with the expected results” (p. 91)
	N/A	15	- “Obtain acquirer (or other authorized stakeholders) acceptance of validation results” (p. 91)
	R, Bs, S, Bs, S, D	1, 2, 5, 6, 8, 9, 17, 12	- “Maintain bidirectional traceability of the validated system elements with the validation strategy, business/mission analysis, stakeholder requirements, system architecture, design, and system requirements.” (p. 91)
	D	12, 17	- “Provide baseline information for configuration management.” (p. 91)
	S, Bs, S, Bs, D	6, 5, 9, 8, 12, 17	- “Update the validation strategy and schedule according to the progress of the project; in particular, planned validation actions can be redefined or rescheduled as necessary.” (p. 91)

Table 16 Models of designing and design issues

	R	F	Be	Bs	S	D
Pahl/Beitz	4	11	17	5	27	17
INCOSE	50	92	137	91	180	105
RUP	2	13	15	10	31	22
DFSS/ICOV	4	11	17	5	27	17

Table 17 Design phases and design issues

	R	F	Be	Bs	S	D
PB—Task clarification	2	4	5	0	0	5
PB—Conceptual design	1	5	10	2	9	2
PB—Embodiment design	1	2	2	2	16	7
PB—Detail design	0	0	0	1	2	3
RUP—Inception	2	4	3	0	0	3
RUP—Elaboration	0	8	5	4	17	9
RUP—Construction	0	1	7	5	13	7
RUP—Transition	0	0	0	1	1	3
DFSS—Identify	2	4	3	0	0	1
DFSS—Conceptualize	0	2	3	1	6	2
DFSS—Optimize	0	0	0	1	2	2
DFSS—Validate	0	0	0	1	2	2
INCOSE—SysReq Def	5	9	10	0	0	4
INCOSE—Arch Def	7	19	21	8	26	7
INCOSE—Design Def	2	5	11	3	15	8
INCOSE—Implementation	2	0	11	1	8	11
INCOSE—Integration	1	5	11	1	12	6
INCOSE—Verification	4	3	7	16	17	13
INCOSE—Validation	5	5	3	17	16	9

Acknowledgements This research is supported by a grant from the US National Science Foundation, grant number CMMI-1762415. 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.

Funding Open access funding provided by Johannes Kepler University Linz.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Asimov W (1962) Introduction to design. Prentice-Hall
- Baines TS et al (2007) State-of-the-art in product-service systems. *Proc Inst Mech Eng Part b J Eng Manuf* 221(10):1543–1552
- Benzecri J-P (2019) Correspondence analysis handbook. Routledge, Oxfordshire
- Blomkvist J (2016) Benefits of service level prototyping. *Des J* 19(4):545–564
- Bott MJ, Mesmer B (2019) Determination of function-behavior-structure model transition probabilities from real-world data. AIAA Scitech 2019 Forum, San Diego
- Cascini G, Fantoni G, Montagna F (2012) Situating needs and requirements in the FBS framework. *Des Stud* 34(5):636–662
- Cavalieri S, Pezzotta G (2012) Product-service systems engineering: state of the art and research challenges. *Comput Ind* 63(4):278–288
- Chang G-S, Perng H-L, Juang J-N (2008) A review of systems engineering standards and processes. *J Biomechatron Eng* 1(1):71–85
- El-Haik B, Roy DM (2005) Service design for six sigma: a roadmap for excellence. John Wiley & Sons, Hoboken
- Finkelstein L, Archer B, Schwarz KK, Constable GEP (1989) The inter-relationship between systems engineering and engineering design. *IEE Proc A Phys Sci Measure Instrum Manage Edu* 136(3):159–164
- Forsberg K and Mooz H (1991) The relationship of system engineering to the project cycle, Proceedings of the National Council for Systems Engineering (NCOSE) Conference, Chattanooga, TN, pp. 57–65.
- Gero JS (1990) Design prototypes: a knowledge representation schema for design. *AI Mag* 11(4):26–36
- Gero JS, Kannengiesser U (2004) The situated function-behaviour-structure framework. *Des Stud* 25(4):373–391
- 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. Springer, pp 263–283
- Gero JS, Kannengiesser U, Pourmohamadi M (2014) Commonalities across designing: empirical results. In: Gero JS (ed) Design computing and cognition'12. Springer, pp 285–302
- Godfrey RM (1990) Some thoughts on engineering systems development. *IEE Proc A Phys Sci Measure Instrum Manage Edu* 137(5):302–308
- Greenacre M (2017) Correspondence analysis in practice, 3rd edn. CRC Press, Boca Raton
- Greene MT, Gonzales R and Papalambros PY (2019) Measuring systems engineering and design thinking attitudes, Proceedings of the 22nd International Conference on Engineering Design (ICED19), Delft, The Netherlands
- Gruber TR (1995) Toward principles for the design of ontologies used for knowledge sharing. *Int J Hum Comput Stud* 43(5–6):907–928

- Hamraz B, Clarkson PJ (2015) Industrial evaluation of FBS Linkage—a method to support engineering change management. *J Eng Des* 26(1–3):24–47
- Hein AM, Poulain B, Jankovic M, Chazal Y and Fakhfakh S (2018) Product service system design in a system of systems context: A literature survey, *Proceedings of the DESIGN 2018 15th International Design Conference*, The Design Society, pp. 2891–2902
- Honour EC (2018) A historical perspective on systems engineering. *Syst Eng* 21(3):148–151
- INCOSE (2015) *Systems engineering handbook: a guide for system life cycle processes and activities*, fourth edition, INCOSE-TP-2003–002–04. International Council on Systems Engineering (INCOSE), San Diego
- ISO/IEC (2002) *PDTR 19760 Systems Engineering – Guide for ISO/IEC 15288 (System Life Cycle Processes)*, ISO/IEC JTC1/SC7 N2683, Technical Report
- ISO/IEC/IEEE (2011) *Systems and Software Engineering – Architecture Description*, ISO/IEC/IEEE 42010:2011(E)
- ISO/IEC/IEEE (2015) *Systems and Software Engineering – System Life Cycle Processes*, ISO/IEC/IEEE 15288:2015(E)
- ISO/IEC/IEEE (2018) *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*, ISO/IEC/IEEE 29148:2018(E)
- Kan JWT, Gero JS (2009) Using the FBS ontology to capture semantic design information in design protocol studies. In: McDonnell J, Lloyd P (eds) *About designing: analysing design meetings*. Taylor and Francis, New York, pp 213–229
- Kannengiesser U, Gero JS (2015) Is designing independent of domain? Comparing models of engineering, software and service design. *Res Eng Design* 26(3):253–275
- Kannengiesser U, Gero JS (2017) Can Pahl and Beitz' systematic approach be a predictive model of designing? *Design Science* 3:E24
- Kasser JE (2020) *Systems engineering: a systemic and systematic methodology for solving complex problems*. CRC Press, Boca Raton
- Kruchten P (2004) *The rational unified process: an introduction*. Addison-Wesley, Upper Saddle River
- Li M (2002) Fostering design culture through cultivating the user-designers' design thinking and systems thinking. *Syst Pract Act Res* 15:385–410
- Maier MW (1998) *Architecting principles for systems-of-systems*. *Syst Eng* 1(4):267–284
- Meyn SP, Tweedie RL (2009) *Markov chains and stochastic stability*. Cambridge University Press, New York, Cambridge
- Mikusz M (2014) Towards an understanding of cyber-physical systems as industrial software-product-service systems. *Procedia CIRP* 16:385–389
- MIT (2003) *ESD Symposium Committee Overview: Engineering Systems Research and Practice*, Working Paper ESD-WP-2003–01.20, Massachusetts Institute of Technology, Engineering Systems Division, Massachusetts, MA.
- NASA (2007) *Systems Engineering Handbook*, NASA/SP-2007-6105 Rev1. National Aeronautics and Space Administration, Washington DC
- Pahl G, Beitz W (2007) *Engineering design: a systematic approach*. Springer, Berlin
- Patel S, Mehta K (2017) Systems, design, and entrepreneurial thinking: comparative frameworks. *Syst Pract Act Res* 30:515–533
- Pauwels P, Strobbe T, De Meyer R (2015) Analysing how constraints impact architectural decision-making. *Int J Design Sci Technol* 21(1):83–111
- Pourdehnad J, Wexler ER and Wilson DV (2011) Systems and design thinking: a conceptual framework for their integration, *Proceedings of the 55th Annual Meeting of the ISSS*, July 17–22, 2011, Hull, UK
- Reich Y, Hatchuel A, Shai O, Subrahmanian E (2012) A theoretical analysis of creativity methods in engineering design: casting and improving ASIT within C-K theory. *J Eng Des* 23(2):137–158
- SAE (1999) *Processes for Engineering a System*, EIA-632, SAE International
- Song T (2014) Expert Vs. Novice: Problem Decomposition/Recomposition in Engineering Design, *PhD Thesis*, Utah State University, Logan, Utah
- VDI (2004) *Design methodology for mechatronic systems*, VDI 2206. Verein Deutscher Ingenieure, Düsseldorf
- von Bertalanffy L (1968) *General system theory: foundations, development, applications*. Braziller, New York
- Watson MD (2019) Systems engineering principles and hypotheses. *Insight* 22(1):18–28
- Wynn DC, Clarkson PJ (2018) Process models in design and development. *Res Eng Design* 29(2):161–202

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.