Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2015), *29*, 33–44.

doi:10.1017/S0890060414000316

Empirical support for problem–solution coevolution in a parametric design environment

RONGRONG YU,[1] NING GU,[1] MICHAEL OSTWALD,[1] AND JOHN S. GERO[2]

[1]School of Architecture and Built Environment, University of Newcastle, Callaghan, New South Wales, Australia
[2]School of Architecture and Department of Computer Science, University of North Carolina at Charlotte, Charlotte, North Carolina, USA

(Received May 21, 2013; Accepted January 9, 2014)

Abstract

This paper describes the results of a protocol study exploring problem–solution coevolution in a parametric design environment (PDE). The study involved eight participants who completed a defined architectural design task using Rhino and Grasshopper software: a typical PDE. The method of protocol analysis was employed to study the cognitive behaviors that occurred while these designers were working in the PDE. By analyzing the way in which the designers shifted between "problem" and "solution" spaces in the PDE, characteristics of the coevolutionary design process are identified and discussed. Results of this research include two potentially significant observations. First, the coevolution process occurs frequently within the design knowledge level (i.e., when using Rhino) and within the rule algorithm level (i.e., when using Grasshopper) of the parametric design process. Second, the designers' coevolution process was focused on the design knowledge level at the beginning of the design session, while they focused more on the rule algorithm level toward the end of the design session. These results support an improved understanding of the design process that occurs in PDEs.

Keywords: Designers' Cognitive Behavior; Parametric Design Environments; Problem–Solution Coevolution; Protocol Analysis

1. INTRODUCTION

Architectural design is not a linear process; it involves the stages of proposition, testing, refinement, analysis, and rejection, but not necessarily sequentially. For example, in a study of Frank Gehry's office, Boland et al. (2008) describe Gehry's design process as existing in a "liquid" state for a long period before eventually becoming "frozen" into a proposition for a building. During the liquid state, drawings and models are made, tested, and rejected, being refined until a "final" solution is crystalized. In this way, Gehry et al. explore and respond to different aspects of the design problem, alternatively shifting the focus from formal solutions to contextual problems, technological challenges, and functional optimization. For Gehry's team, this shift from problem definition and analysis to solution proposition and testing is often signaled by the decision to digitize a physical model for further refinement and development. While Gehry's forms and buildings may appear to be more challenging than those of many other architects, his team follows a much more common cyclical design process, which reflects architects' thoughts, actions, and behaviors as they shift their focus from considering design problems to testing design solutions. The parallel development of problems and solutions in this "liquid state," which many architects follow, is called coevolution.

The concept of the coevolution of a design "problem space" and "solution space" has been proposed by Maher and Poon (1996). Design is a process that develops or formulates a problem and ideas for a solution, in parallel (Dorst & Cross, 2001). It has even been suggested that this coevolution of design problem and solution spaces is an indicator of creativity in design (Maher & Tang, 2003). However, the medium or environment in which the design process is undertaken (be it physical and sketch based, or digital and CAD based) also has a significant effect on designers' cognitive processes (Chen, 2001; Mitchell et al., 2003). One particular medium, the parametric design environment (PDE), has become increasingly prevalent in architectural design in recent years. According to Kolarevic (2003), the change in designing associated with parametricism is characterized by a rejection of the static solutions offered in conventional design systems and the adoption of intelligent systems. This change is claimed to have rendered design processes both more flexible

Reprint requests to: Rongrong Yu, School of Architecture and Built Environment, University of Newcastle, University Drive, Callaghan, NSW 2308, Australia. E-mail: rongrong.yu@uon.edu.au

and more productive. Various studies support this view, arguing that parametric tools can advance design processes in a variety of ways (Qian et al., 2007; Schnabel, 2007; Abdelmohsen & Do, 2009; Iordanova et al., 2009). However, there is a lack of empirical evidence supporting an understanding of designers' behavior in the PDE. This paper focuses on exploring the coevolution of design problem and solution spaces in PDEs.

In order to examine the way designers think and act in a PDE, the designers' cognitive activities while they are designing in a PDE are studied. In the study reported here, eight designers were asked to complete an architectural design task in a PDE. Protocol analysis (Ericsson & Simon, 1993; Gero & McNeill, 1998) was then employed to examine the designers' behavior and identify the cognitive activities associated with problem–solution coevolution in PDEs.

2. BACKGROUND

2.1. Parametric design

Parametric design is a digital design method that is characterized by rule algorithm design and multiple solution generation (Abdelsalam, 2009; Karle & Kelly, 2011). As Woodbury (2010) argues, it supports the creation, management, and organization of complex digital design models. The term "parameters" is used to describe factors that determine a series of variations leading to a potentially infinite range of possibilities being generated (Kolarevic, 2003). In the architectural design industry, parametric design tools are mainly used for complex form generation, multiple design solution optimization, as well as structure and sustainability control. Common parametric design software includes Generative Component from Bentley Corporation, Digital Project from Gehry Technology, and Grasshopper from McNeel. Scripting tools, for authoring and defining parameters, include Processing based on the Java language, Rhino script, and Python script, based on VB language from McNeel. In this study, we chose Grasshopper as the PDE because it is an advanced environment for facilitating conceptual design and a relatively popular tool in the architectural profession.

Researchers have suggested that parametric tools support the design process in various ways. For instance, Iordanova et al. (2009) examined students' behavior when using parametric design software and found that ideas were generated more rapidly in PDEs, while more variations also emerged simultaneously. Schnabel (2007) argues that PDEs are beneficial for generating unpredicted events and for accommodating changes. However, researchers have typically studied design behavior in PDEs by using informal observation techniques and interviews with students in a studio or workshop setting. Such approaches typically lack empirical evidence and can provide only a very limited understanding of designers' behaviors. This empirical gap is addressed in the present study by adopting the method of protocol analysis, a method used for in-depth or detailed analysis of any number of cases or participants. Lee et al. (2012) have demonstrated the use of protocol analysis to evaluate creativity in PDEs. Using the same method, Chien and Yeh (2012) explore "unexpected outcomes" in PDEs. Results of these studies both confirm the viability of the method and also suggest that some conditions in PDEs can potentially benefit the designers' process.

2.2. Problem–solution coevolution in design

Design is not just a linear process of finding solutions to an initial given task or requirement; it is also about redefining and reframing the design problems that have been provided (Asimov, 1962; Schön & Wiggins, 1992; Suwa et al., 2000). During the design process, designers continue to redefine their design intentions, searching for alternative resolutions. This iterative process, which revisits both problems and solutions during the design process, should not simply be regarded as a cyclical series of events, because with each recursion in the process, the parameters have evolved and shifted. Previous studies show that the expert design process also involves a close interaction between representations of problems and solutions (Cross, 2011). The coevolution of design problem and solution spaces is one possible way to conceptualize the design process. Instead of seeing design as a process of progressive refinement (concept design, to schematic design, to developed design), design could be analyzed through the way the cognitive effort shifts between the consideration of problems and solutions (Maher & Poon, 1996; Dorst & Cross, 2001). Design is a process that uses analysis, synthesis, and evaluation as it shifts between the design problem and possible solutions (Asimov, 1962; Lawson, 1997; Cross, 2011). It is during this process that designers formulate critical questions and explore answers, and thus the developing relationship between the "problem space" and the "solution space" is at the core of the coevolution model of design. Maher and Poon (1996) and Dorst and Cross (2001) each use this model to suggest that the coevolution of design problem and solution spaces has a close correlation with the occurrence of design creativity.

The concept of problem–solution coevolution is described by Maher and Poon (1996; Fig. 1). Although it was developed for computational exploration (from an AI perspective), the model also describes a common design process. In the coevolution model, the problem space (P) and solution space (S) in-

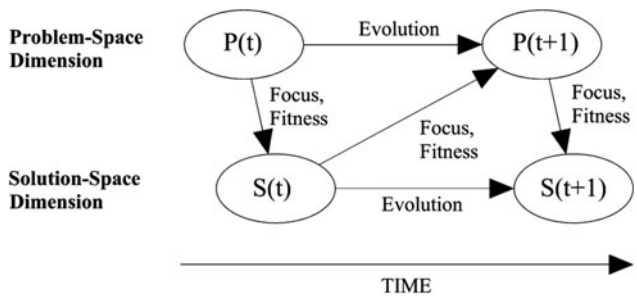

Fig. 1. The coevolution model based on Maher and Poon (1996).

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

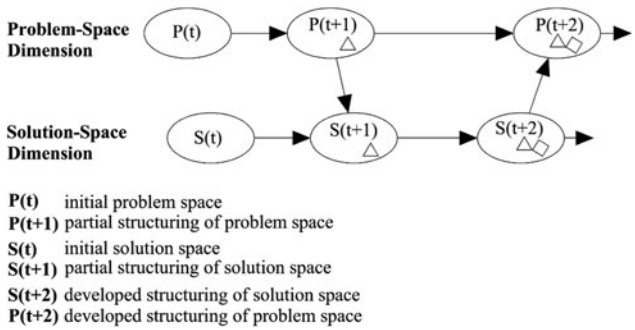

Fig. 2. The coevolution model based on Dorst and Cross (2001).

teract over time (t; Fig. 1). Designers start by analyzing the initial design requirements and formulating the design problem, $P(t)$. While exploring possible design solutions $S(t)$ for the problem $P(t)$, new intentions are added into the problem space over time $P(t + 1)$. This is a core process for coevolution in design, and particularly so when the solution does not satisfy a key requirement; by changing or adapting the requirements and intentions, a satisfactory problem and solution pair could be generated (Maher & Poon, 1996; Dorst & Cross, 2001).

In a development of Maher and Poon's coevolution model, Dorst and Cross (2001) propose a refined version that further illustrates the creative process from a behavioral perspective. Their study employed the method of protocol analysis, in which nine industrial designers were observed. In their model (Fig. 2), the designers start from a design problem space $P(t)$, and develop a partially structured problem $[P(t + 1)\triangle]$, which is then used to develop a partially structured solution space $[S(t + 1)\triangle]$ of $S(t)$. This process is repeated throughout the design progress, as Maher and Tang (2003) suggest, with the transition between design problem and solution occurring in cyclical iterations until a satisfactory solution is developed. Dorst and Cross further argue that this coevolution process is vital to support the highest level of creative design (Cross & Cross, 1998; Dorst & Cross, 2001). While the focus of the present paper is not explicitly on creativity in the design process, there are, as this past research suggests, several indicators of creative potential in the coevolutionary design process, which the present research can consider in the context of PDEs.

3. APPLYING THE FUNCTION–BEHAVIOR–STRUCTURE (FBS) ONTOLOGY TO EXPLORE THE COEVOLUTION PROCESS IN PDES

3.1. Protocol studies using the FBS ontology

Protocol studies record and convert qualitative verbal and gestural utterances and actions into quantitative research data (Ericsson & Simon, 1993; Gero & Mc Neill, 1998). They have been used extensively in design research to develop an understanding of design cognition (Suwa & Tversky, 1997; Atman et al., 1999; Kan & Gero, 2008). In such studies a common coding scheme used for design analysis is based on the FBS ontology (Gero, 1990; Gero & Kannengiesser, 2004), which has since been applied to a variety of studies on designers' cognitive behavior (Gero & McNeill, 1998; Kan & Gero, 2009; Lammi, 2011; Tang et al., 2011; Kan & Gero, 2012).

The FBS ontology (Fig. 3) contains three classes of concepts: function (F), behavior (B) and structure (S). Function represents the intentions or purposes of the design; behavior represents how the structure of a designed artifact achieves its functions, either derived (Bs) or expected, from (Be) structure; and structure represents the components that make up an artifact and their relationships. Figure 3 identifies and numbers the eight design processes derived from this ontology: formulation, analysis, evaluation, synthesis, documentation, and three types of reformulation (–1, –2, and –3). Among the eight design processes, the three types of reformulation are said to be the dominant ones, potentially capturing innovative or creative aspects of designing by introducing new variables or directions (Kan & Gero, 2008). The present study of coevolution in PDEs adopts the FBS ontology as its central theoretical framework because it clearly distinguishes the eight types of design processes, which provides opportunities to examine design cognition in detail (Kan & Gero, 2009). Kan and Gero (2012) apply the FBS ontology to a study of software designers' behavior, suggesting that the method was effective for encoding programming/rule-based activities across different de-

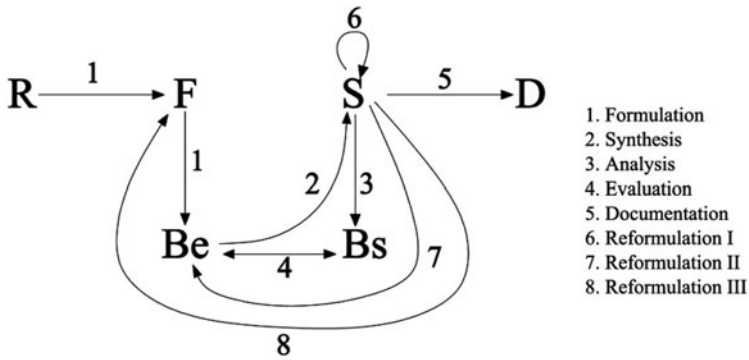

Fig. 3. The function–behavior–structure ontology based on Gero and Kannengiesser (2004).

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

sign disciplines. Given that PDEs enable scripting/programming activities, it is anticipated that the FBS scheme will be able to encode both geometric modeling and rule-based algorithmic activities effectively. For this study, one of the original codes in the FBS ontology, description (*D*), has been excluded because in PDEs this process rarely occurs. For example, whereas in the sketching environment there is a regular need to document or describe design decisions and actions (*D*), parametric modeling tools automatically document and describe a three-dimensional model directly from the actions of programming and scripting and as part of the consideration of structure (*S*). Therefore, the present study does not include a consideration of the description (*D*) code.

3.2. Two levels of design activities in PDEs

The ways in which parametric design is used by architects are not well understood, which is why some argue that parametric design "requires a deeper understanding of how it can support our intentions as architects" (Sanguinetti & Kraus, 2011, p. 47). Compared to traditional design environments, in PDEs architects not only design by applying specialist knowledge but also define rules and their logical relationships using parameters (Abdelsalam, 2009). Woodbury (2010) claims that in the PDE, designers need a different kind of geometric knowledge that can "predict persistent effects to understand the diversity and structure of the mathematical toolbox, and to shuttle between the intended effect and mathematical invention that models it." This implies that designers need to know how the mathematical tools will work in the design development processes in addition to their base architectural knowledge. As Aranda and Lasch (2008) observe, the parametric design process mediates between two worlds. The first is an abstract and coded system from which complex spatial forms emerge through rule-based mathematical expressions. The second world is the space where the designers apply their design knowledge to address the needs of people, cultures, communities, and cities. When an architect models a building form using parameters, he must assess variations, design data flow routes, and adjust the values of parameters, and revise rules. At this time, the architect is thinking about not only the particular building design but also the rule design. It is through the control of logical relationships between forms and functions that the possibilities for design solutions are heightened (Hernandez, 2006; Karle & Kelly, 2011).

Therefore, in a typical parametric design process, there are design activities on two levels: the design knowledge level and the rule algorithm level. In the design knowledge level, architects make use of their design knowledge, including, for example, how to adapt a building to the site, how to shape the way people use the building, and how to satisfy the requirements of their clients. At the rule algorithm level, designers apply design knowledge through the operations of the parametric design tools, including defining the rules and their logical relationships, choosing the parameters suitable for a particular purpose, and importing external data into the proposed rules. During the design process, designers progress by applying specialist knowledge; in some parts (viz., the rule algorithm level), they apply design knowledge indirectly by defining rules and their logical relationships, and this is known as parameterization. In order to capture the processes involved in designing in PDEs, the main class of variables from the original FBS ontology that have been used, which are also referred to as cognitive design issues or just design issues, are *R*, *F*, *B*e, *B*s, and *S*. Each variable is then further decomposed into the two levels of design activities: the design knowledge level, denoted by the superscript K, and the rule algorithm level, denoted by the superscript R (Fig. 4). This does not require an extension of the ontological variables because each decomposition is an instantiation of the base ontological variable.

3.3. Interpretation of FBS coding in a PDE

1. *Requirement (R^R) in the rule algorithm space:* Requirement (*R*) variables include "all requirements and constraints that were explicitly provided to the designers at the outset of the design task" (Gero et al., 2014, p. 286). Within the context of the present study, requirement (R) codes refer to these moments when designers considered or reviewed the content of the design brief provided. Because there is only design related information in the brief, all the requirement variables will be coded as R^K. Therefore, there will be no instances of "requirement" variables at the rule algorithm level (R^R).
2. Function (F^R) in the rule algorithm space: In the FBS ontology, function (*F*) variables describe "the teleology of the object, which means 'what it is for'" (Gero & Kannengiesser, 2004, p. 374). Function *F* is the purpose or intention of a design, which shapes the idea in designers' cognitive thinking. The concept of *F* does not vary between different design environments. The claim is that design tools do not affect the "function" of the design. In the PDE, the architect still needs to consider design intention and decide which factors to parameterize or constrain and where to assign the weight for specific factors (Ottchen, 2009, p. 23). Therefore, in PDEs F variables comply with the original understanding of function in the FBS model. Thus, if designers talk about the design intention or purpose, the segments should be coded as "function" (F^K). When designers talk about function of the rule, it is about the effect they want the rule to achieve. These segments are coded as expected behavior of the rule (Be^R). Therefore, there will be no instances of "function" variables at the rule algorithm level (F^R).

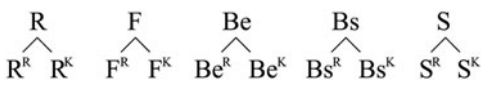

Fig. 4. Applying the function–behavior–structure ontology in the parametric design environment.

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

3. *Behavior (B) in the rule algorithm space:* In the FBS ontology, behavior (*B*) variables "describe the attributes that are derived or expected to be derived from the structure variables of the object, which means 'what it does'" (Gero & Kannengiesser, 2004, p. 374). There are two types of *B*: expected behavior (*B*e) and behavior derived from structure (*B*s). In PDEs, *B* variables express different meanings at the rule algorithm level.
   - *B*e: A *B*e is one where "designers use theory or experience to speculate what effect could fulfill a purpose before a specific structure is proposed" (Jiang, 2012, pp. 36–37). This interpretation has been well understood at the design knowledge level. When it comes to the rule algorithm level, expected behavior of the rule (Be^{R}) means that designers set up algorithm goals or think about the way to achieve those goals in the rule algorithm space (Table 1).
   - *B*s: A Bs is an actual behavior. At the design knowledge level, Bs represents an evaluation of existing geometry/structure, while at the rule algorithm level, Bs signifies an evaluation of the structure of the rule algorithm. When designers examine the current rule, the segments will be coded as "Bs^{R}" (Table 2).
4. *Structure (S) in the rule algorithm space:* In the FBS ontology, structure (*S*) variables describe "the components of the object and their relationships, which mean 'what it is'" (Gero & Kannengiesser, 2004, p. 374). In the design knowledge space, structure (*S*) variables refer to the elements or relationships of the geometries, whereas in the rule algorithm space, it is defined as the structure of the rule algorithm: the components of rules and their relationships for parameterization. In PDEs, designers also produce form as geometry, that is, structure in the design knowledge space. This can be modeled by directly applying design knowledge, or through the rule algorithm. In the latter case, a set of parameters and parametric commands will be used. Designers define relationships to connect these elements to form the rule algorithm. When designers organize the structure of rules or make parametric commands, the segments will be coded as S^{R} (Table 3).

Table 1. *Expected behavior interpretation in the rule algorithm space (Be^{R})*

Description	Typical Activities
Set up rule algorithm goals	"The points will be generated randomly."
Ways to achieve certain rule algorithm goals or related actions	"The façade is divided up into the panels." "I will try to get these intermedium points and create a line."
Utilizing mathematically related commands for parameterization	Set Grasshopper components such as domain, function, graft, and flatten.

Table 2. *Structure behavior interpretation in the rule algorithm space (Bs^{R})*

Description	Typical Activities
Checking data flow routes	Set "panel" or other Grasshopper components for checking.
Evaluating existing rules	Make a judgment about the rule: "it has some problem" etc. or check the rule definition from the scripting interface.

3.4. Problem–solution division in the PDE

This paper adopts the problem and solution division identified in the FBS model (Jiang et al., 2014) to distinguish problem-related and solution-related design issues. In the FBS ontology, problem-related issues include design consideration about requirements (*R*), function (*F*), and expected behaviour (*B*e). Solution-related issues involve design considerations about structure (*S*) and behaviour derived from the structure (*B*s).

Based on Jiang et al.'s problem and solution division, Table 4, and the understanding of the two levels of activities in the PDE (Fig. 4), design problem-related issues and design solution-related issues are further divided into: problem-related issues at knowlege level (P_K, including R^K, F^K, and Be^K), problem-related issues at the rule algorithm level (P_R, including Be^R); solution-related design issues at the knowlege level (S_K, including Bs^K and S^K) and solution-related design issues at the rule algorithm level (S_R, including Bs^R and S^R), as shown in Table 5. This division is consistent with the

Table 3. *Structure interpretation in rule algorithm space (S^{R})*

Description	Typical Activities
Talking about or making the structure of rules	Connect/organize rule components
Applying geometry-making commands with features for parameterization	Set/change parameters, set/change relationships, etc.

Table 4. *Mapping function–behavior–structure design issues and processes onto problem and solution spaces (Jiang et al., 2014)*

Problem/Solution Space	Design Issue
Problem space = problem-focused design issues	Requirement (*R*) Function (*F*) Expected behavior (*B*e)
Solution space = solution-focused design issues	Behavior derived from structure (*B*s) Structure (*S*)

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

Table 5. *Mapping function–behavior–structure design issues onto problem and solution spaces in the parametric design environment*

Problem/Solution Space	Design Issue
Reasoning about problem at design knowledge level (P_K)	Requirement (R^K)
	Function (F^K)
	Expected behavior (Be^K)
Reasoning about problem at rule algorithm level (P_R)	Expected behavior (Be^R)
Reasoning about solution at design knowledge level (S_K)	Behavior derived from structure (Bs^K)
	Structure (S^K)
Reasoning about solution at rule algorithm level (S_R)	Behavior derived from structure (Bs^R)
	Structure (S^R)

characteristics of parametric design in that designers not only think about problems from the design perspective but also have to formulate problems using rule design, which is essential in the PDE.

4. EXPERIMENT SETTING

The study reported in this paper is based on the work of eight designers, who are all professional architects with an average of 8 years of experience, and with no less than 2 years of experience using parametric design. In the experiment, each designer was required to complete a defined architectural design task in a PDE. During the experiment, both designers' activities and their verbalizations ("thinking aloud") were video-recorded including a screen capture program and the recorded data subsequently used for the protocol analysis. The design environment was Rhino and Grasshopper, a typical PDE. Designers were given 40 min for the design session, along with time to allow for familiarization with the particular systems being used. The design task was a conceptual design for a commercial building containing specific functions and located on a premodeled site (Fig. 5), which was provided to each designer. Because this study focused on exploring designers' behavior at the conceptual design stage, participants were required to only consider concept generation, simple site planning, and general functional zones. No detailed plan layout was required. The tasks were both open and general enough to provide designers with the freedom to enable various possible strategies to be applied during the parametric design process. As a result, designers exhibited their own typical ways of approaching parametric design so that some of our findings may begin to be generalized. During the experiment, designers were not allowed to sketch manually, so almost all their actions occurred on the computer to ensure that the design environment was purely within the PDE. All of these controls were put in place to limit any variables that could potentially bias the study.

5. EXPLORING THE IMPACT OF RULE ALGORITHMS ON THE COEVOLUTION PROCESS IN THE PDE

5.1. General results

In order to produce robustness of the protocol coding results, two rounds of segmentation and coding were conducted with an interval of 2 weeks between them. Following the two segmentation and coding rounds, an arbitration session was carried out to produce the final protocol from the combination of the two rounds. The percentage of agreement between the two rounds was 83.4%, ($SD = 5.7\%$) and between the individual rounds and the final arbitrated results, 91.5% ($SD = 3.1\%$). These percentages are indicative of the methodological reliability of the coding process and results. Analysis of the eight

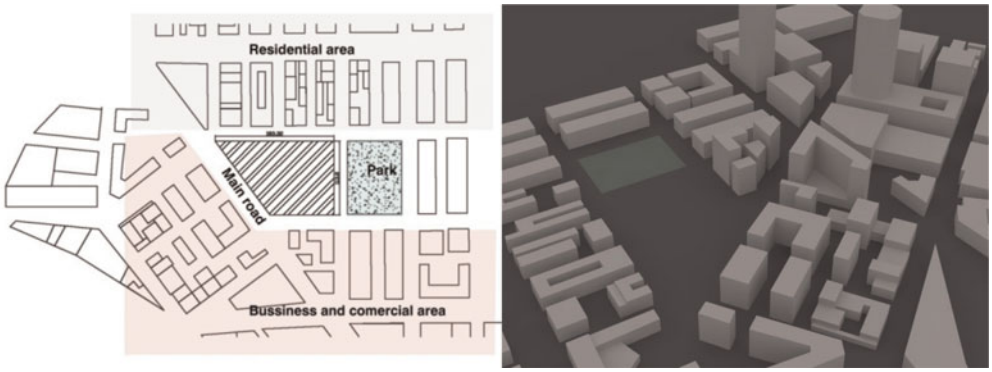

Fig. 5. Site model provided to the designers during the experiments.

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

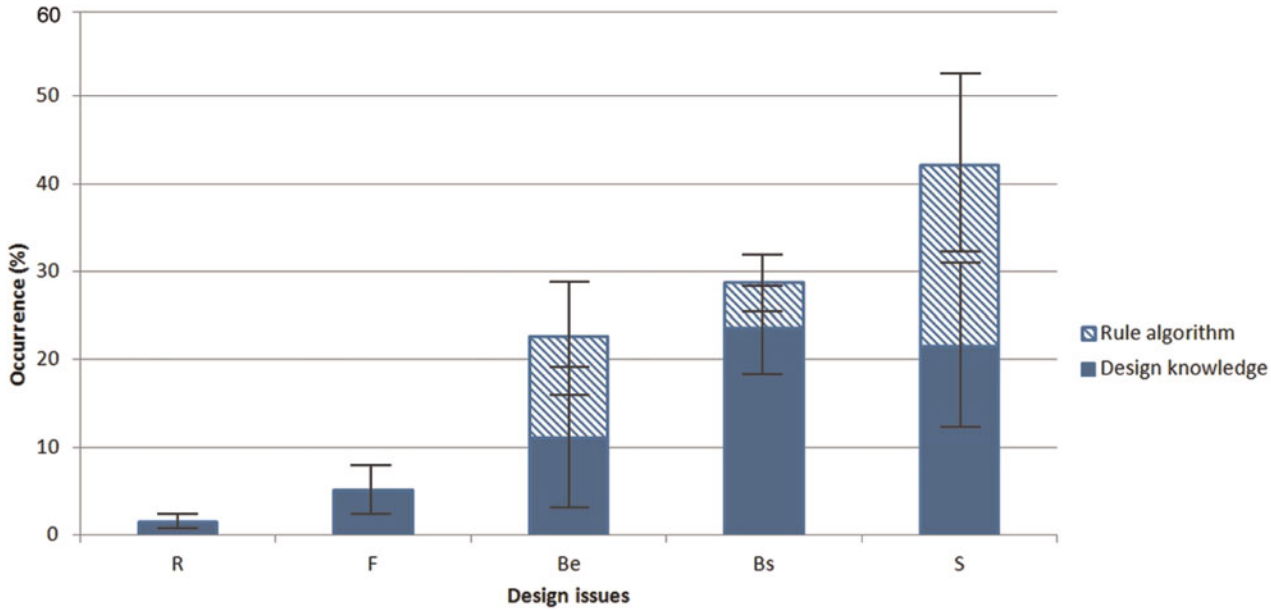

Fig. 6. Design issue distribution at both the design knowledge and rule algorithm levels in the parametric design environment.

protocols revealed the mean total numbers of segments was 244 ($SD = 29.7$, where a segment is the division of protocols into individual units; in the current research, the division was based on the FBS ontology and coresponded to design issues). The mean time spent on the design sessions was 48.4 min ($SD = 7.4$), and over 92.2% of all segments were coded using the FBS model. Noncoded segments include those concerned with communication and software management.

The distributions of "design issues" in the protocols, both at the knowledge and the rule algorithm levels in the PDE, are shown in Figure 6. From Figure 6 it can be seen that solution-related design issues (*B*s and *S*) have the highest frequency, and there are fewer problem-related issues (*R*, *F*, and *B*e). Qualitatively, the rule algorithm plays an important part in both *B*e and *S*. Particularly from the *B*e result, we can infer that designers set rule algorithm goals/requirements and consider the way to achieve the rule algorithm goals frequently in the PDE. The quantitative anlysis of FBS design issue occurrences are shown in Table 6. The occurrence of design issues was normalized by dividing them by the total number of coded design issues. The results indicate that designers spent most cognitve effort on the structure-related design issues, which consist of S^{K} (21.56%) and S^{R} (20.49%). These are followed by Bs^{K} (23.55%) and Bs^{R} (5.18%). Considerably less effort was spent on R^{K} (1.52%) and F^{K} (5.09%).

5.2. Transition patterns between the design problem and solution spaces in the PDE

The nature of the transitions between problem space and solution space, which is critical to the concept of coevolution (Dorst & Cross, 2001), can be examined by calculating the discontinuity ratio of the designers' design process, a value which can indicate the frequency of interactions between design problem and solution spaces. The discontinuity ratio is the ratio of the number of transitions to the overall number of segments [Eq. (1)]. This ratio represents the frequency of transitions between the problem and solution spaces in a given period. The higher values of this ratio indicate that interactions between design problem space and solution space occur more frequently, which suggests a productive coevolution process.

$$\text{discontinuity ratio} = \frac{\sum_{\text{transition number}}}{\sum_{\text{overall segments number}}} \times 100\%. \quad (1)$$

The discontinutiy ratios of transition between the design problem and solution spaces in the PDE are shown in Figure 7. The numbers on the arrows represent the average (of eight participants) discontinuity ratio of each transition during the entire design process. For instance, for designer

Table 6. *Design issue analysis*

	R^{K} (%)	F^{K} (%)	Be^{K} (%)	Be^{R} (%)	Bs^{K} (%)	Bs^{R} (%)	S^{K} (%)	S^{R} (%)
Mean	1.52	5.09	11.08	11.59	23.55	5.18	21.56	20.49
SD	0.72	2.81	7.96	6.12	5.09	2.80	9.37	10.18

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

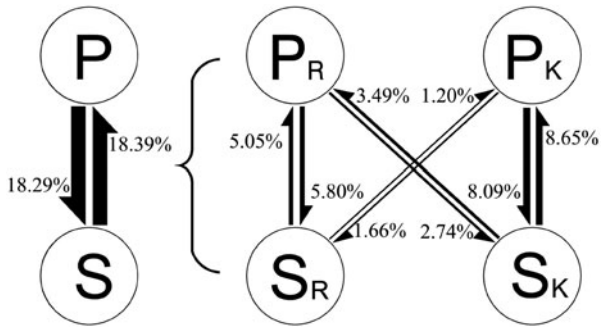

Fig. 7. Discontinuity ratios between the design problem and solution spaces.

E, the number of transitions from P_R to S_K is 11, the overall coded segment number is 243, thus the discontinuity ratio of P_R to S_K is 11/243 = 4.53%. Table 7 shows the quantitative analysis of the transitions between design problem space and design solution space.

As shown in Figure 7 and Table 7, the discontinuity ratios of transition from design problem space to solution space (18.29%) and from design solution to problem space (18.39%) are similar. Within this the dominant ones are between P_R and S_K (with discontinuity ratios of 8.65% and 8.09%), and between P_R and S_R (with discontinuity ratios of 5.80% and 5.05%). From this we can infer that the transitions tend to remain within the design knowledge level or the rule algorithm level and less frequently occurs across different levels. Although the transition across different levels occurs less frequently, among the transitions across different levels, there are relatively more between P_R and S_K (with discontinuity ratios of 2.74% and 3.49%), which might mean that designers sometimes reframe the design problem space or requirements at the rule-algorithm level based on design-knowledge-related considerations. One example of S_K to P_R that occurs frequently in the PDE is that designers set new rule-related design goals based on the evaluation of the geometry model at the design knowledge level. The pattern that most infrequently appears is the transition between P_K and S_R (with discontinuity ratios of 1.66% and 1.20%). In particular, there is only a very small percentage for the occurrence of the transition from S_R to P_K (with discontinuity ratio of 1.20%), which suggests that designers rarely reframe design problems at the design knowledge level based on the rule algorithm solution.

5.3. Transition patterns across the whole design session

In order to further articulate the eight types of transitions (as outlined in Fig. 7 and Table 7) between the problem space and solution space across the design session, the distribution of the discontinuity ratio of each transition in the PDE is presented in Figure 8. The horizontal axis in Figure 8 is the design session divided into ten subsessions, deciles, each with an equal number of segments, while the vertical axis represents the average discontinuity ratio (of the eight participants) of the transition patterns in each decile of the design session.

In the following description, we define the "early design stage" as the period 1–3.3 on the horizontal axis, the "mid-design stage" as 3.4–6.7, and the "end design stage" as 6.7–10. The descriptors are thus time based, rather than a direct indicator of the degree to which a design has been completed.

The eight types of transitions between the design problem space and design solution space are shown by the eight lines in Figure 8, respectively, representing P_R to S_R, P_K to S_K, P_R to S_K, P_K to S_R, S_R to P_R, S_K to P_R, S_K to P_R, and S_R to P_R. At the early design stage the dominant transition is between P_K and S_K. At the mid-design session, the dominant transition is between P_R and S_R, although the transition from S_K to P_K is still active. There are more transitions between P_R and S_R toward the end of the design session. We can infer from this that, at the beginning of the design session, the coevolution process is focused on the design knowledge level; at the mid-design session, the coevolution process is active at both design knowledge and rule algorithm levels; and at the end of the design session, it is more focused on the rule algorithm level. The reason for this pattern may be that at the beginning designers considered the brief from the design knowledge perspective, which is similar to common architectural practice. Later, designers started using the rule algorithm process to implement their goals or concepts. During this process, designers continued to redefine the design problem while they were searching for solutions at a design knowledge level. This is supported by observations of the experiment, which noted that designers tended to start from the brief, then analyze the site, and then develop basic concepts. In the next stage, designers started considering the form or structure of their design, and the rules to implement them. That is, they set rule algorithm goals and explored different ways to achieve them. Meanwhile, designers constantly returned to the design knowledge level to evaluate the current design, and in this

Table 7. *Transition occurrences between design problem space and design solution space*

	Transition From Problem to Solution				Transition From Solution to Problem			
	P_R–S_R (%)	P_K–S_K (%)	P_R–S_K (%)	P_K–S_R (%)	S_R–P_R (%)	S_K–P_K (%)	S_K–P_R (%)	S_R–P_K (%)
Mean	5.80	8.09	2.74	1.66	5.05	8.65	3.49	1.20
SD	3.17	1.28	0.46	0.21	1.20	1.29	0.61	0.19

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

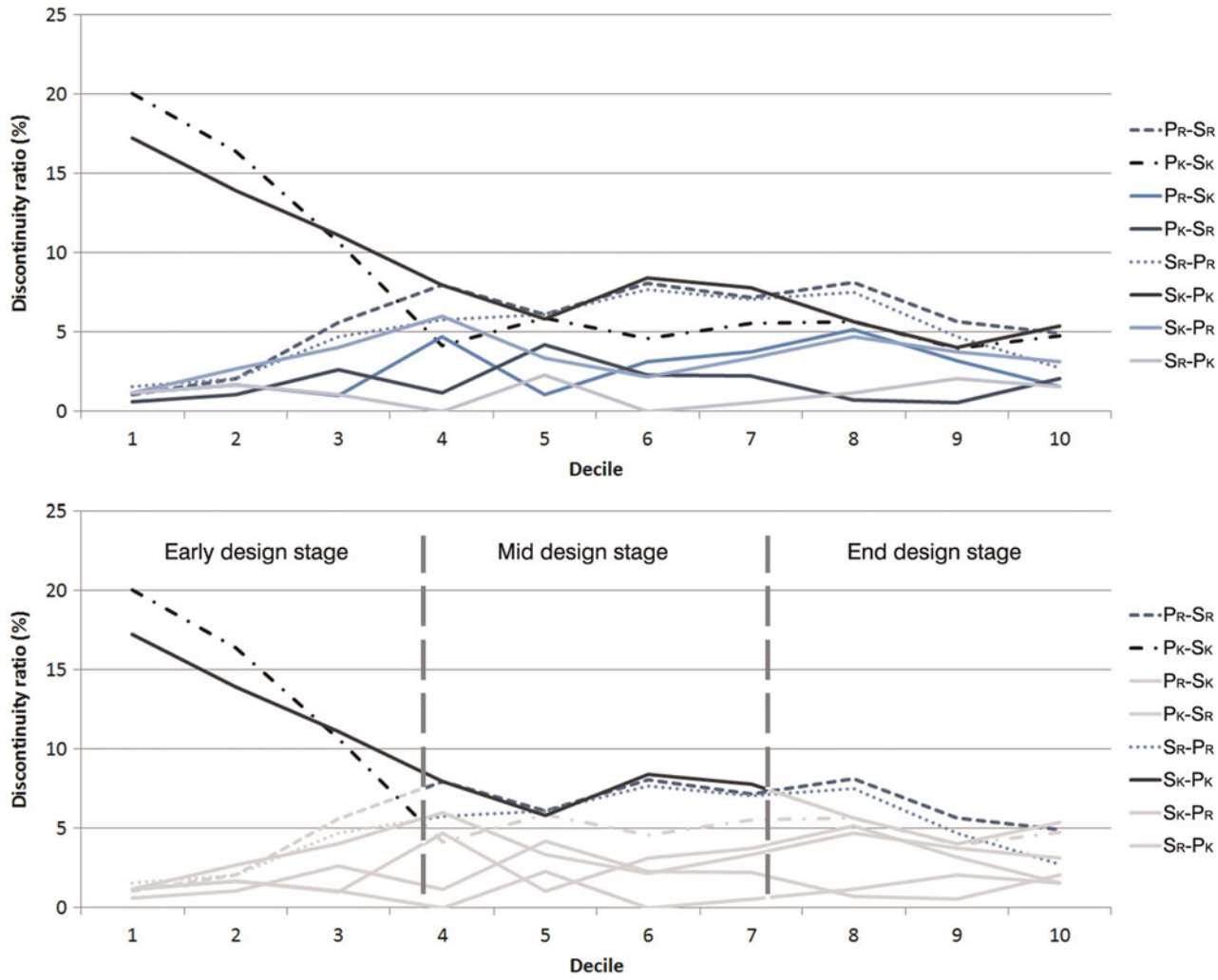

Fig. 8. Distribution of the discontinuity ratios in the parametric design environment across time.

way, the initial concept was developed and evolved gradually. At the end of the session, designers concentrated on the rule algorithm design and tried to finalize the model using rules.

5.4. A model of coevolution process in the PDE

The transition processes between design problem and solution spaces at both the design knowledge level and rule algorithm level are shown in Figure 9. The horizontal moves indicate the problem space (P) evolving from time t to time $t + 1$. The vertical moves are processes where "the problem leads to the solution" or "the solution refocuses the problem" (Maher & Poon, 1996). These moves comply with Maher and Kundu's (1993) finding that design requirements would change with the design solution: the solution space $S(t)$ is not only a space where a design solution can be explored but also prompts new require-

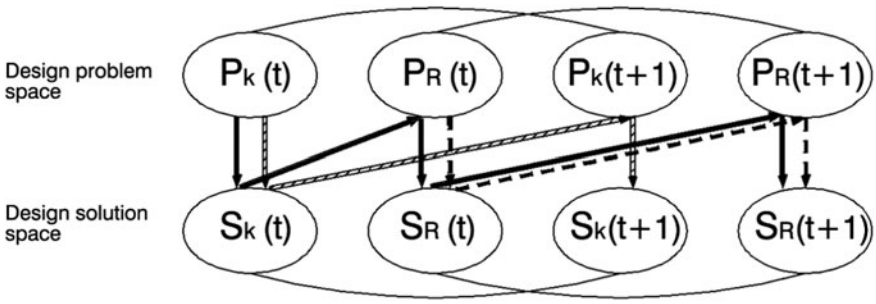

Fig. 9. A model of the coevolution process in the parametric design environment.

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

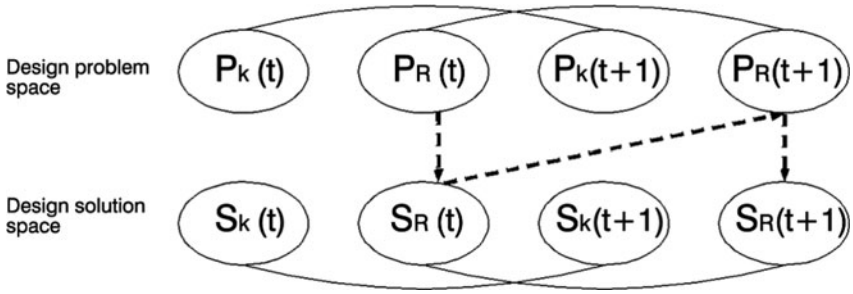

Fig. 10. The coevolution process at the rule algorithm level.

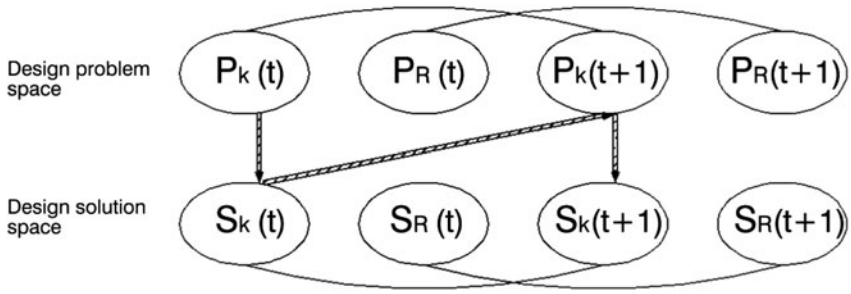

Fig. 11. The coevolution process at the design knowledge level.

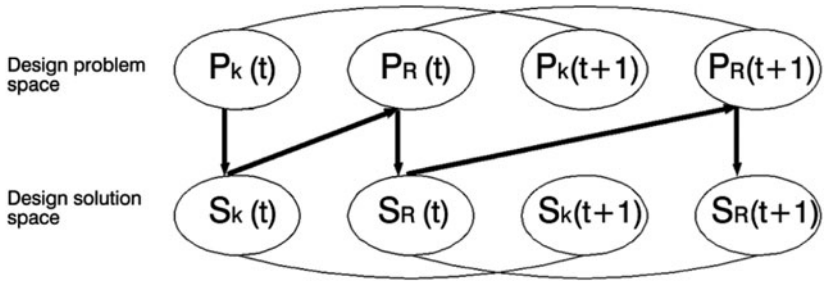

Fig. 12. The coevolution process across the design knowledge level and the rule algorithm level.

ments in $P(t+1)$ that were not in the original problem space $P(t)$. Figure 9 illustrates a model of the coevolution process in the PDE as identified from this study. The details of this model are further articulated in Figures 10, 11, and 12.

1. Figure 10 presents the coevolution of the problem and solution spaces at the rule algorithm level (indicated as dashed arrows). This coevolution process frequently occured in the PDE. Designers explored solutions for the rule algorithm goals/requirements, $P_R(t)$, from the solution space $S_R(t)$; they refined or added new requirements to reformulate the rule algorithm problem $P_R(t+1)$.
2. Figure 11 presents the coevolution of the problem and solution spaces at the design knowledge level (indicated as solid dashed arrows). This is the most frequently occurring coevolution process in the PDE. This behavior is similar to that in traditional design environments (Maher & Poon, 1996; Dorst & Cross, 2001).
3. Figure 12 presents the coevolution process across the design knowledge level and the rule algorithm level (indicated as solid arrows). In this coevolution process, designers started from the problem space at the design knowledge level, $P_K(t)$. During the exploration in the solution space $S_K(t)$, there were new requirements emerging at the rule algorithm level. The design problem space at the rule algorithm level $P_R(t)$ was refined. Then the exploration of a design problem and solution changed the direction to the rule algorithm level. This is a process in the PDE, in which designers explore the design solution and reformulate the problem across the design knowledge level and the rule algorithm level.

6. Conclusion

Design may be conceptualized as a special class of problem-solving processes (Simon, 1969) where the problems are either not clearly defined (Maher et al., 1996; Chi, 1997) or

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

ill defined (Simon, 1973; Corne et al., 1994). This is why, in a design process, designers constantly return to the design problem space to reformulate the challenge they are facing (Simon, 1973). Through the interaction between cognitive activities in the design problem and solution spaces, the design is progressed until a "satisfactory" outcome is identified (Maher & Tang, 2003). As Cross (2011) and Schön (1983) have suggested, creative design is a process of exploration; during the process, problem and solution spaces are evolving and unstable until fixed by an "emergent" bridge, or a satisfactory problem–solution pair. This coevolution process is significant for understanding the design process.

Parametric design differs from design that uses traditional geometrical modeling because it is reliant on rule algorithms that must operate in parallel with other traditional design behaviors (Yu et al., 2012). In this paper, we have studied the coevolution process in the PDE by examining empirical data derived from experiments with professional designers. The data was generated by employing the protocol analysis method. Through this study the parametric design process has been categorized by a two-level model of design activities: design knowledge and rule algorithm. From the results of the experiment, this division is capable of capturing parametric design behaviors in a sufficiently comprehensive manner that they can help us to understand the design process in this environment.

Based on the division of design activities into two levels, and by calculating the frequency of transitions between the design problem and solution spaces, three particular characteristics of the coevolution process in the PDE have been identified. The first of these is that the coevolution process typically occurs at the individual design knowledge level or rule algorithm level, and only relatively infrequently do transitions occur across the two levels. Second, the designers' coevolution process is focused on the design knowledge level at the early design stage, while they use more cognitive effort at the rule algorithm level toward the end of the design session. Those activities that are, therefore, most representative of operations in the PDE (activities on the rule algorithm level) play a more important role in the later stages of the session than in its earlier stages. Third, a model that illustrates the main coevolution process in the PDE has been proposed. In this model, three main coevolution subprocesses are identified: coevolution at rule algorithm level, coevolution at the design knowledge level, and coevolution across the rule algorithm and design knowledge levels. The proposed model assists in formally understanding designers' behavior in terms of interaction between problems and solutions when designing in PDEs.

REFERENCES

Abdelmohsen, S., & Do, E.Y.-L. (2009). Analyzing the significance of problem solving expertise and computational tool proficiency in design ideation. *Proc. Int. Conf. Computer Aided Architectural Design Futures, CAADFutures 2009*, pp. 273–287. Montréal: Presses de l'Université de Montréal.

Abdelsalam, M. (2009). The use of the smart geometry through various design processes: using the programming platform (parametric features) and generative components. *Proc. Int. Conf. Arab Society for Computer Aided Architectural Design, ASCAAD 2009*, pp. 297–304. Manama, Bahrain: Arab Society for Computer Aided Architectural Design.

Aranda, B., & Lasch, C. (2008). What is parametric to us. In *From Control to Design: Parametric/Algorithmic Architecture* (Sakamoto, T., & Ferré, A., Eds.), pp. 194–206. Barcelona, Spain: Actar-D.

Asimov, W. (1962). *Introduction to Design*. Upper Saddle River, NJ: Prentice–Hall.

Atman, C.J., Chimka, J.R., Bursic, K.M., & Nachtmann, H.L. (1999). A comparison of freshman and senior engineering design processes. *Design Studies 20(2)*, 131–152.

Boland, R.J., Collopy, F., Kalle, L., & Youngjin, Y. (2008). Managing as designing: lessons for organization leaders from the design practice of Frank O. Gehry. *Design Issues 24(1)*, 10–25.

Chen, S.-C. (2001). The role of design creativity in computer media. *Proc. Int. Conf. Education and Research in Computer Aided Architectural Design in Europe, eCAADe 2001*, pp. 226–231. Helsinki: eCAADe in cooperation with Mediatecture.

Chi, M.T.H. (1997). Quantifying qualitative analyses of verbal data: a practical guide. *Learning Science 6(3)*, 271–315.

Chien, S.-F., & Yeh, Y.-T. (2012). On creativity and parametric design—a preliminary study of designer's behaviour when employing parametric design tools. *Proc. Int. Conf. Education and Research in Computer Aided Architectural Design in Europe, eCAADe 2012*, pp. 245–253. Prague: eCAADe in cooperation with Mediatecture.

Corne, D., Smithers, T., & Ross, P. (1994). Solving design problems by computational exploration. In *Formal Design Methods for Computer-Aided Design* (Gero, J., & Tyugy, N., Eds.), pp. 249–270. Amsterdam: Elsevier.

Cross, N. (2011). *Design Thinking: Understanding How Designers Think and Work*. New York: Berg.

Cross, N., & Cross, C. (1998). Expertise in engineering design. *Research in Engineering Design 10(3)*, 141–149.

Dorst, K., & Cross, N. (2001). Creativity in the design process: coevolution of problem–solution. *Design Studies 22(5)*, 425–437.

Ericsson, K.A., & Simon, H.A. (1993). *Protocol Analysis: Verbal Reports as Data*. Cambridge, MA: MIT Press.

Gero, J.S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine 11(4)*, 26–36.

Gero, J.S., & Kannengiesser, U. (2004). The situated function–behaviour–structure framework. *Design Studies 25(4)*, 373–391.

Gero, J.S., Kannengiesser, U., & Pourmohamadi, M. (2014). Commonalities across designing: empirical results. *Proc. Design Computing and Cognition '12* (Gero, J.S., Ed.), pp. 285–302. Berlin: Springer.

Gero, J.S., & McNeill, T. (1998). An approach to the analysis of design protocols. *Design Studies 19(1)*, 21–61.

Hernandez, C.R.B. (2006). *Design procedure: a computational framework for parametric design and complex shapes in Architecture.* PhD Thesis. Cambridge, MA: MIT.

Iordanova, I., Tidafi, T., Guité, M., De Paoli, G., & Lachapelle, J. (2009). Parametric methods of exploration and creativity during architectural design: a case study in the design studio. *Proc. Int. Conf. Computer Aided Architectural Design Futures, CaadFutures 2009*, pp. 423–439. Montréal: Presses de l'Université de Montréal.

Jiang, H. (2012). *Understanding senior design students' product conceptual design activities —a comparison between industrial and engineering design students.* PhD Thesis. Singapore: National University of Singapore.

Jiang, H., Gero, J.S., & Yen, C.C. (2014). Exploring designing styles using a problem–solution index. *Proc. Design Computing and Cognition '12* (Gero, J.S., Ed.), pp. 85–101. Berlin: Springer.

Kan, J.W.T., & Gero, J.S. (2008). Acquiring information from linkography in protocol studies of designing. *Design Studies 29(4)*, 315–337.

Kan, J.W.T., & Gero, J.S. (2009). Using the FBS ontology to capture semantic design information in design protocol studies. In *About Designing: Analysing Design Meetings* (McDonnell, J., & Lloyd, P., Eds.), pp. 213–229. New York: Taylor & Francis.

Kan, J.W.T., & Gero, J.S. (2012). Studying software design cognition. In *Software Designers in Action: A Human-Centric Look at Design Work* (Petre, M., & van der Hoek, A., Eds.), p. 61–77. Abingdon: Chapman & Hall/CRC.

Karle, D., & Kelly, B. (2011). Parametric thinking. *Proc. Int. Conf. Parametricism (SPC) ACADIA Regional 2011*, Paper No. 109, Lincoln, NE, March 11–12.

https://doi.org/10.1017/S0890060414000316 Published online by Cambridge University Press

Kolarevic, B. (2003). *Architecture in the Digital Age: Design and Manufacturing*. New York: Spon Press.
Lammi, M. (2011). *Characterizing high school students' systems thinking in engineering design through the function–behavior–structure (FBS) framework*. PhD Thesis. Logan, UT: Utah State University.
Lawson, B. (1997). *How Designers Think: The Design Process Demystified*. Oxford: Architectural Press.
Lee, J.H., Gu, N., Jupp, J., & Sherratt, S. (2012). Evaluating creativity in parametric design processes and products: a pilot study. *Proc. Int. Conf. Design Computing and Cognition, DCC'12*. College Station, TX: Springer .
Maher, M.L., & Kundu, S. (1993). Adaptive design using a genetic algorithm. In *Formal Design Methods for Computer-Aided Design* (Gero, J.S., & Sudweeks, F., Eds.), pp. 240–248. Sydney: University of Sydney, Key Centre of Design Computing.
Maher, M.L., & Poon, J. (1996). Modelling design exploration as coevolution. *Microcomputers in Civil Engineering 11(3)*, 195–210.
Maher, M.L., Poon, J., & Boulanger, S. (1996). Formalising design exploration as coevolution: a combined gene approach. In *Advances in Formal Design Methods for CAD* (Gero, J.S., & Sudweeks, F., Eds.), pp. 3–30. London: Chapman & Hall.
Maher, M.L., & Tang, H.H. (2003). Coevolution as a computational and cognitive model of design. *Research in Engineering Design 14(1)*, 47–63.
Mitchell, W.J., Inouye, A.S., & Blumenthal, M.S. (2003). *Beyond Productivity: Information Technology, Innovation, and Creativity*. Washington, DC: National Academies Press.
Ottchen, C. (2009). The future of information modelling and the end of theory: less is limited, more is different. *Architectural Design 79*, 22–27.
Qian, C.Z., Chen, V.Y., & Woodbury, R.F. (2007). Participant observation can discover design patterns in parametric modeling. *Proc. Int. Conf. Association for Computer Aided Design in Architecture, ACADIA2007*, pp. 230–241. Halifax, NS: Riverside Architectural Press and Tuns Press.
Sanguinetti, P., & Kraus, C. (2011). Thinking in parametric phenomenology. *Proc. Int. Conf. Parametricism (SPC) ACADIA Regional 2011*, Paper No. 39, Lincoln, NE, March 11–12.
Schnabel, M.A. (2007). Parametric designing in architecture. *Proc. Int. Conf. Computer Aided Architectural Design Futures, CAADFutures 2007*, pp. 237–250. Sydney: Springer.
Schön, D.A., & Wiggins, G. (1992). Kinds of seeing and their functions in designing. *Design Studies 13(2)*, 135–156.
Schön, D.A. (1983). *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books.
Simon, H.A. (1969). *The Sciences of the Artificial*. Cambridge, MA: MIT Press.
Simon, H.A. (1973). The structure of ill-structured problems. *Artificial Intelligence 4(3–4)*, 181–204.
Suwa, M., Gero, J.S., & Purcell, T. (2000). Unexpected discoveries and S-invention of design requirements: important vehicles for a design process. *Design Studies 21(6)*, 539–567.
Suwa, M., & Tversky, B. (1997). What do architects and students perceive in their design sketches? A protocol analysis. *Design Studies 18(4)*, 385–403.
Tang, H.H., Lee, Y.Y., & Gero, J.S. (2011). Comparing collaborative co-located and distributed design processes in digital and traditional sketching environments: a protocol study using the function–behaviour–structure coding scheme. *Design Studies 32(1)*, 1–29.
Woodbury, R. (2010). *Elements of Parametric Design*. New York: Routledge.
Yu, R., Gu, N., & Ostwald, M.J. (2012). Using situated FBS ontology to explore designers' patterns of behavior in parametric envrionments. *Journal of Information Technology in Construction 17*, 271–282.

Rongrong Yu is a PhD candidate in the School of Architecture and Built Environment at the University of Newcastle. Her PhD study is currently focused on exploring designers' behavior in parametric design. Her research interests include computer-aided design, architecture design theory, design cognition. She has published journal and conference papers related to the topic.

Ning Gu is a Senior Lecturer at the School of Architecture and Built Environment at the University of Newcastle. He currently supervises four PhD students and a number of Honours research students. His research has been in the broad area of design computing and cognition since 1999. His career highlights include leading and participating in various Australian Research Council (ARC) Discovery and Co-operative Research Centre for Construction Innovation projects. The outcomes of his research have been documented in over 130 peer-reviewed journal and conference publications.

Michael J. Ostwald is an ARC Future Fellow and Dean of the School of Architecture and Built Environment at the University of Newcastle. He is a Visiting Professor at RMIT and was previously a Professorial Research Fellow at Victoria University Wellington. Michael is Co-Editor in Chief of the *Nexus Network Journal: Architecture and Mathematics*, and he is on the editorial boards of *ARQ* and *Architectural Theory Review*.

John S. Gero is a Research Professor in architecture and computer science at the University of North Carolina, Charlotte, and a Research Professor in the Krasnow Institute for Advanced Study in Computational Social Science at George Mason University. Previously he was Professor of Design Science and Co-Director of the Key Centre for Design Computing and Cognition at the University of Sydney. He has published over 50 books and 600 research papers. His research has been cited over 4,500 times in the last 5 years.