Research in Engineering Design

DOI 10.1007/s00163-002-0016-y

Original Paper

Co-evolution as a computational and cognitive model of design

Mary Lou Maher(✉) · Hsien-Hui Tang

M.L. Maher
Key Centre of Design Computing and Cognition, University of Sydney, NSW 2006 Australia

H.H. Tang
The Graduate Institute of Architecture, National Chiao Tung University, Hsinchu 30050 Taiwan, hhtang@arch.nctu.edu.tw

✉ E-mail: Mary@arch.usyd.edu.au

Received: 7 September 2001 / **Revised:** 10 April 2002 / **Accepted:** 23 April 2002 / **Published online:**

Abstract. Co-evolutionary design has been developed as a computational model that assumes two parallel search spaces: the problem space and the solution space. The design process iteratively searches each space using the other space as the basis for a fitness function when evaluating the alternatives. Co-evolutionary design can also be developed as a cognitive model of design by characterizing the way in which designers iteratively search for a design solution, making revisions to the problem specification. This paper presents the computational model of co-evolutionary design and then describes a protocol study of human designers looking for evidence of co-evolution of problem specifications and design solutions. The study shows that co-evolutionary design is a good cognitive model of design and highlights the similarities and differences between the computational model and the cognitive model. The results show that the two kinds of co-evolutionary design complement each other, having strengths in different aspects of the design process.

Keywords. Co-evolutionary design - Computational model of design - Cognitive model of design - Protocol studies

1

Introduction

Co-evolution is a term used to identify the process in nature in which two or more species interact so intimately that their evolutionary fitness depends on each other. Biological co-evolution has been the inspiration for a class of computational algorithms called co-evolutionary computing (Paredis *1998*). Co-evolutionary design, as introduced in Maher (*1994*), is an approach to design problem solving in

which the requirements and solutions of design evolve separately and affect each other. The model of co-evolutionary design has been developed as a computational model to show how a mechanism for design can include reasoning about the problem in parallel with reasoning about the design solution. Various versions of a co-evolutionary algorithm and its applications have been developed, and the most recent has been published in Maher (*2001*). We summarize the model here and then show how this model can be used to describe our observations of human design cognition.

Simon (*1969*) characterized design as a search process, allowing the design process to be understood as one of the "sciences of the artificial". Since then the design research community has embraced the "design as search" model by formulating the goals, state spaces, and operators for various design domains and design problems. Although the search paradigm is very strong and still underpins much of problem solving, other models of design have been proposed that address the formalisation of design knowledge and design goals. Here we briefly compare three related models to highlight the characteristics of co-evolutionary design: search, exploration, and co-evolution.

Design can be formalized as search when the goals of the design are well-defined before search commences and the focus of design is not changed until a solution is found. Design becomes exploration when different parts of the solution space are searched or the solution space is expanded through a change in the focus of the design. A focus, or set of requirements, determines which solution in a solution space is considered best. Design is co-evolutionary when there are two distinct spaces being searched, and the problem space and solution space change through mutual interaction. The focus of the search is based on the problem requirements when searching the solution space, and is based on the solutions when searching the problem requirements space.

The model of design as co-evolution is illustrated in Fig. 1. The interactions between two sets of spaces can occur through the change in focus of the search and by passing variables from one space to another. The downward arrow corresponds to the evaluation of *S* using *P* as the source of the focus. If no satisfactory solution is found with the stated design requirements, the solution space becomes the basis to derive the focus for searching the problem space. The upward arrow corresponds to the evaluation of *P* using *S* as the source of the focus. After searching for relevant requirements in the problem space, the new problem requirements space is the basis of the focus for the search for a solution (introduced in Maher and Poon *1996*). The interaction between the spaces can include transferring variables from one space to the other.

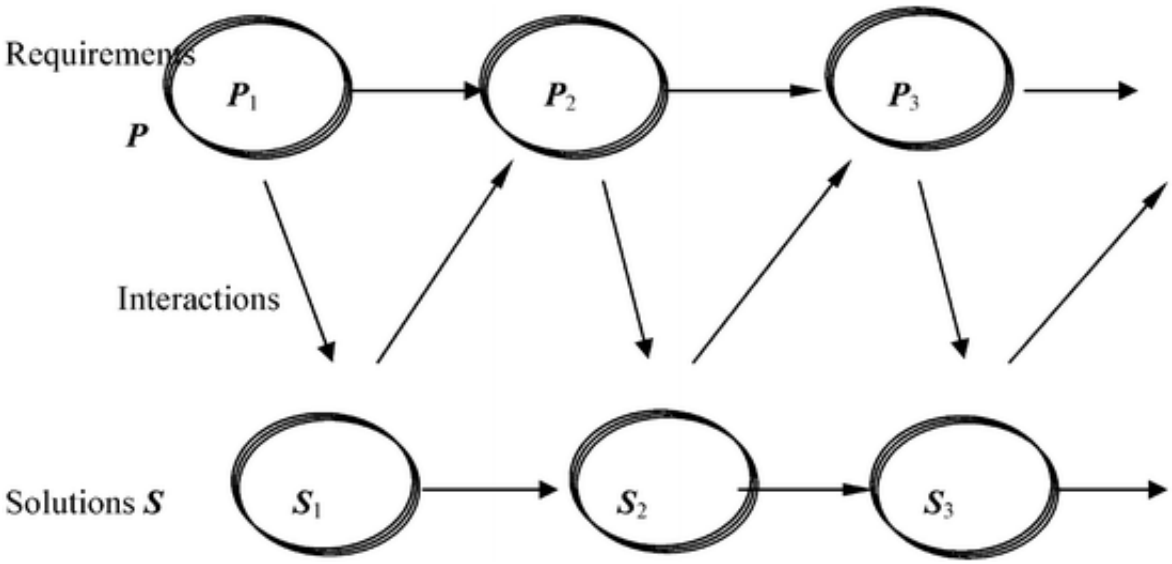

Fig. 1. A model of co-evolutionary design (after Maher *1994*)

Design as co-evolution explores the spaces of problem requirements and design solutions iteratively. Interactions between the spaces may add new variables into both spaces. Usually, the requirements and solution of an ill-defined design problem cannot be understood well enough to define a fixed focus. The search for potential solutions is often interleaved with changes in the requirements for the solution. Figure 2 illustrates the input and output of the co-evolutionary model of design.

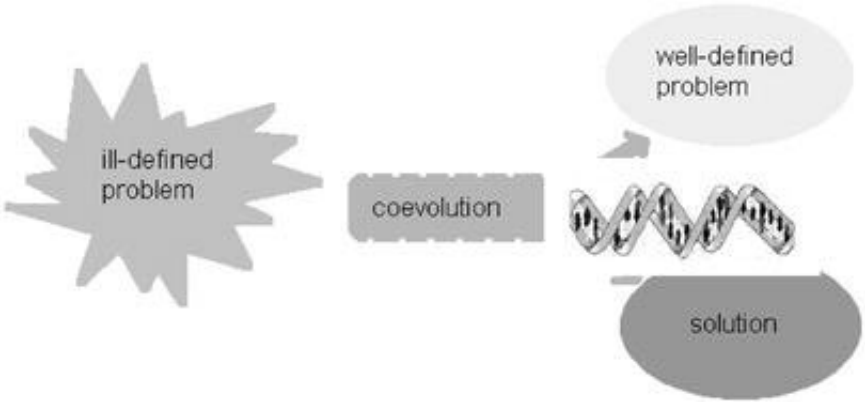

Fig. 2. Input and output of co-evolutionary design

2

Computational co-evolutionary design algorithm

Computational co-evolutionary design is characterized by having a search space of problem requirements and a search space of problem solutions. We use a genetic algorithm (GA) formalism to implement computational co-evolutionary design using concepts of search spaces, genotype/phenotype representation of individuals in the search spaces, and the operators of crossover, mutation, selection, reproduction, and fitness evaluation. In the first phase of co-evolutionary design, the problem space provides the basis for a fitness function used to evaluate alternatives in the design space. In the second phase, the solution space provides the basis for a fitness function used to evaluate the problem space. Each phase is essentially a goal-directed search process with a fixed goal.

First, the requirements space is used to define an initial focus, as illustrated in Fig. 3. The solution space is a population of alternative designs. Applying the evolution operators, the members of the population in the solution space go through repeated cycles of selection, reproduction, crossover, and mutation. At each generation, the solutions are evaluated against a fitness function. When the process has converged, and if the process has not yet reached the termination condition, the focus shifts and the search changes to the problem space.

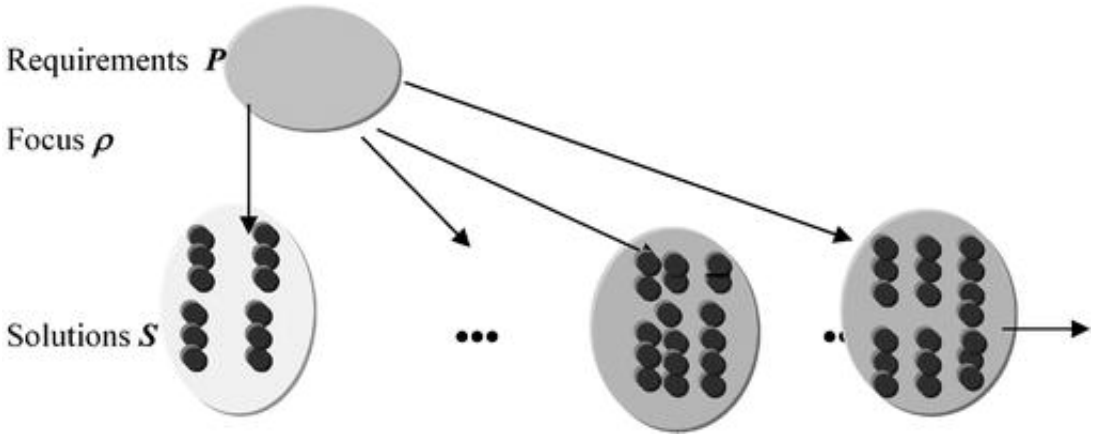

Fig. 3. Evolutionary search for design solutions

After convergence, the solution space is used to develop a focus for searching the problem space, as illustrated in Fig. 4. The search space is a population of alternative requirements. Applying the evolution operators, the members of the population in the problem space go through repeated cycles of selection, reproduction, crossover, and mutation. At each generation, the requirements are checked for convergence. When the process has converged, the focus shifts and the search changes to the solution space. Logically, termination is not possible at this phase because this phase results in a new problem definition and does not necessarily have a corresponding design solution.

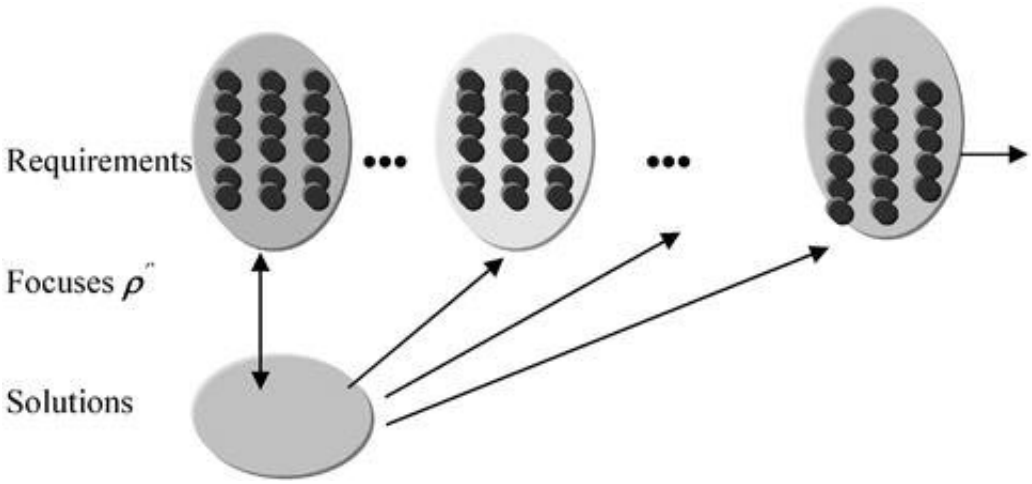

Fig. 4. Evolutionary search for design requirements

A general algorithm of co-evolutionary design is shown below. The algorithm is adapted and extended from a simple GA (for example, see Mitchell *1996*). Each of the two phases of co-evolutionary design is a search process using a simple GA and unchanging fitness function, denoted ρ . Therefore, each

phase corresponds to one design focus and a change in phase indicates a change in focus. The different evolutionary cycles are illustrated graphically in Fig. 5. The concept of a "generation" or "cycle" in co-evolution has different meanings from a simple GA since there is more than one space that is evolving. Therefore, the algorithm has three counters for time: *T*, *t*, and *v*. The quantity *T* indicates the change in generation from one full cycle of co-evolution to another; *T* is updated after the solution space and the problem space have been searched and converged. The quantity *t* indicates a new generation in the simple GA sense. Therefore *t* is updated after each generation of search in one of the spaces; that is, after a cycle of selection, crossover, and mutation. The quantity *v* is similar to *t*, except that the value is reset for each phase so that *v* is a local counter for evolutionary cycles.

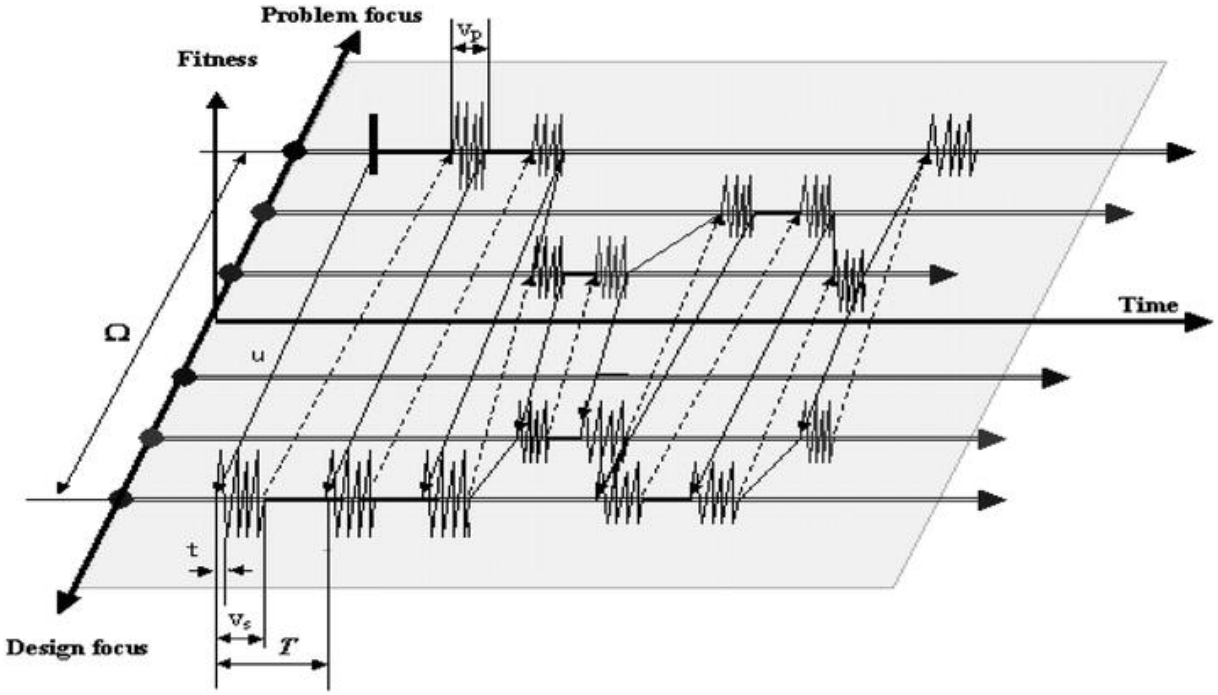

Fig. 5. Illustration of the co-evolutionary design algorithm

/ CoDESIGN algorithm */.*

```
T=0;      /*counter for co-evolutionary cycles*/
t=0;      /*counter for evolutionary cycles*/
Ω ={};     /*set of fitness functions*/
```

initialize genotypes of requirements P_t and solutions S_t;

While termination conditions, $f(T,\Omega)$, are not satisfied

```
   T=T+1;
   t=t+1;
```

/*Phase 1: determine focus for new problem requirements, redefine problem requirements space, search for best problem requirements*/

```
   If T≠ 1 then
      v=1;          /*counter for local evolutionary cycles*/
      ρ'_{t,T}=f(S_t);   /*fitness function*/
```

$\Omega = \Omega \cup \{\rho'_{t,T}\}$ /*history of fitness functions*/

$P_t = g(P_t, \rho'_{t,T})$ /*revised problems requirements space*/

Repeat

$t = t+1$;

$v = v+1$;

P_t ::=select genotypes in P_{t-1};

reproduce, crossover, and mutate genotypes in P_t;

calculate fitness of phenotypes in P_t using ρ'_T;

Until convergence by similarity of members of population: $h(v, P_t, P_{t-1})$;;

/*Phase 2: determine focus for new design solution, redefine design solution space, search for best design solution*/

$v = 1$; /*counter for local evolutionary cycles*/

$\rho_{t,T} = f(P_t)$; /*fitness function*/

$\Omega = \Omega \cup \{\rho_{t,T}\}$ /*history of fitness functions*/

$S_t = g(S_t, \rho_{t,T})$ /*revised design solution space*/

Repeat

$t = t+1$;

$v = v+1$;

S_t ::=select genotypes in S_{t-1};

reproduce, crossover, and mutate genotypes in S_t;

calculate fitness of phenotypes in S_t using ρ_T;

Until convergence by similarity of members of population: $h(v, S_t, S_{t-1})$;

3

Observing design using protocol studies

The co-evolutionary model of design can also be the basis for a cognitive model suggesting that human designers show evidence of modifying the problem statement and design solution in an iterative and interactive way. By making this assumption, we do not mean that human designers use the operators of a genetic algorithm, that is, selection, crossover, and mutation, but that human designers modify and shift the range of problem requirements and design solutions over time in a way that resembles the co-evolutionary model as illustrated in Fig. 1. The purpose of this study is to assume a co-evolutionary model as the basis for a cognitive model and compare the computational and cognitive models.

We look for features of the co-evolutionary model in human design processes in two experiments using protocol analysis. In this section we provide an overview of the methodology of design protocol studies as a way of observing co-evolutionary design in humans before presenting the empirical results of the experiments. Two specific studies are described: a kettle design and an architectural design. With the kettle design experiment we have developed a qualitative analysis of the evolutionary changes in design requirements, design solutions, and their interactions. With the architectural design experiment we have developed a more quantitative analysis of the cycles of evolution and co-evolution.

Protocol studies and applied concurrent protocol analysis were first introduced in 1970 to research the human design process (Eastman *1970*). Eastman showed the diversity of constraints and the significance of representational language in intuitive design. Since then, the design community has published various protocol studies in different design disciplines. The essential ones include mechanical engineering (Stauffer and Ullman *1991*), software design (Guindin *1990*), electrical design (McNeill et al. *1998*), industrial design (Cross et al. *1996*), architecture (Akin *1993*), and interior design (Eckersley *1988*). This phenomenon indicates that protocol analysis has been accepted as a prevailing approach to elucidating the design process in design studies.

Protocols, the main material analyzed in protocol analysis, are the audio and video information recorded while subjects are designing. They are used to understand different aspects of the design process, and to explore cognitive activities in this process. Most protocol studies use concurrent protocols, meaning the subjects talk aloud, think, and sketch simultaneously while designing. However, whether talking aloud influences the subject’s perception and thinking process is a controversial issue even with deliberated methodology and painstaking experimental procedure. Ericsson and Simon (*1993*) reviewed relevant studies and claimed that if studies meet the criteria for procedure, giving concurrent protocols does not change the end products of thoughts, except by increasing the solution time because of the verbalization. Their strong arguments still could not entirely resolve doubts about the validity of protocol analysis, but in design studies the focus has shifted from the validity of the methodology to the insufficiency of think-aloud to reveal perceptual and insight aspects of design activities. Concurrent protocols are accepted as valid data in revealing the process-oriented aspects of the design process in comparison to the content-oriented aspects (Gero and Tang *2001*).

Recently, the content-oriented protocol study, which isd the second type of protocol studies, has been recognized (Dorst and Dijkhuis *1995*) and largely applied (Suwa et al. *1998*, *2000*; Suwa and Tversky *1997*; Tang and Gero *2001a*, *2001b*). Generally, studies applying concurrent protocols focus on the process-oriented aspects of designing, since they are largely based on the view of information processing proposed by Simon (*1992*). This category of research emphasizes design problems, design

strategies, and issues derived from the design process (Akin *1993*; Dorst *1996*; Eckersley *1988*; Gero and Tang *1999*). In contrast, studies utilizing retrospective protocols are appropriate when studying the cognitive content of the design process, since they are founded on the notion of reflection of actions proposed by Schön (*1983*). They focus on the internal cognitive behaviors to understand the meaning of a designer's drawing, seeing, and issues derived from design content (Suwa et al. *1998*, *1999*; Suwa and Tversky *1997*). In this study, we apply the concurrent protocol in the kettle study and the retrospective protocol in the architectural study using available experimental data.

Generally, there are five steps to a protocol analysis: 1) conducting experiments; 2) transcribing protocols; 3) parsing segments; 4) encoding raw protocols by the coding scheme; 5) analyzing encoded protocols. The difference between concurrent and retrospective protocols are in reporting timing: in the former, subjects report and design simultaneously; while in the latter designers design without interference and then report with the aid of videotapes recording their design sessions. Among these steps, the segmentation and the coding scheme play essential roles in manipulating protocols. The selection of a segmentation method and a coding scheme are influenced by the particular theory or model of design being explored. In our case they were influenced by the co-evolutionary model of design.

4

Electric kettle design experiment

4.1

Experimental setting

The kettle design experimental data was taken from another protocol study reported in Gero and McNeill (*1998*) and McNeill et al. (*1998*). Voluntary participants were recruited from third-year students at the Faculty of Design, Architecture and Building in University of Technology, Sydney. Devices used were two video recorders, one tie-clipped microphone with one amplifier, one TV, and one VCR. Recorders were arranged to cause minimum interference in subjects. They respectively provided a microscopic view, showing what was written and sketched on the papers, and a macroscopic view, showing what gestures the subject presented during the process. The experiments took place at one of their studios to decrease the tensions caused by experiments.

The instructions play a very important role in directing subjects to give protocols in an appropriate way. The final version includes warm-up exercises, main experiments, and a brief interview. Following the instructions, subjects were asked to talk aloud all details of what they thought while solving the design problem, which is documented in the design brief. The experimenter tried to avoid interfering with subjects, and subjects were supposed to think aloud instead of explaining their design concepts to the experimenter. The experimenter sat behind the subjects and kept silent, unless obvious pauses in speeches appeared for a noticeable duration or subjects required relevant information about the design.

4.1.1

The design brief

The design task for this experiment was to design new electrical kettles for a famous international company. The main design point was to develop the interface of kettles for preventing accidental scalds. Subjects were asked to consider both functions and forms of the kettle, developing a concept design within 40 min. This design problem was selected to be a size that could be addressed in the time given and to be to effective in revealing the design process.

4.1.2

Segmentation

After transcribing the talking in the video data, the raw protocol is parsed into small units called segments. The purpose of segmentation is to facilitate the analysis process because the encoding method applies a code to a single segment. One segment may belong to one or more of the subclasses in one predetermined category, depending on the coding scheme that is applied. In recent protocol studies, the protocol was divided along lines of designers' intentions and actions, instead of time unit, verbalization events, or syntactic markers. Therefore, the designer's intention is interpreted in each utterance, and each segment represents one single intention of the designer during the design process.

Although the definition of one segment appears precise and clear in the literature, it is still ambiguous when applied in practice. The methods of segmentation in two recent papers (Gero and Mc Neill *1998*; Suwa et al. *1998*) are claimed to be similar to Goldschmidt's definition (*1994*); however, in fact they are different to some extent. For example, the relationship between segments and the encoding categories is not similar. In the former paper, one encoding subcategory corresponds to one segment, while in the latter paper, there may be more subcategories corresponding to one segment. Moreover, the bases of segmentations are different. The segmentation proposed by Gero and McNeill (*1988*) are principally based on the transcripts, while the segmentation proposed by Suwa et al. (*1998*) are essentially based on the designer's actions in the video. As a result, the meanings of segmentations are different in these papers despite their claims. In this experiment, we segmented the data according to the designer's intention and modified the method corresponding to our content.

4.1.3

Coding scheme

The coding schemes in most previous protocol studies are closely related to information processing theory. The obvious example was Eastman's study (*1970*). He used design units, constraints, and manipulations to encode protocols, and then tabulated the problem behavior graph representing the design process in terms of information processing. Similarly, Chan (*1990*) used a more detailed coding scheme to analyze the architectural design process. Akin (*1984*) conducted the first protocol studies focusing on architectural design in terms of the concept of plan.

Recent protocol research has identified two significant coding schemes for different types of protocols (Gero and McNeill *1998*; Suwa et al. *1998*). Gero and McNeill propose a coding scheme to understand the process-oriented aspects of design. It consists of the problem domain and the design strategy related to the design process. In contrast, the action categories proposed by Suwa and Tversky (*1997*) were established to understand the content-oriented aspects of design. They used notions proposed by Larkin and Simon (*1987*) to define three subclasses for analyzing what designers saw and possibly thought. The coding schemes were further developed by Suwa et al. into very detailed action categories to understand the roles of sketches and unexpected discovery (Suwa et al. *1998*, *2000*).

These two coding schemes provided influential advances in protocol analysis, and this study adapts part of these results.

Here, we establish our own coding scheme in order to observe the human design process as a co-evolutionary process. The first level of coding is to encode the segments into problem requirements (*P*) and solutions (*S*), the two spaces in the computational model of co-evolutionary design. The two spaces are characterized by the role of each search space in the design process: the problem requirements space defines the features and constraints that specify required or desired aspects of a design solution, and the design solution space defines the features and behaviors of a range of design solutions. Therefore, the segments were encoded into four categories:

- Features of problem requirements (*P*-fea): as declarative statements about specific features required in the design solution;
- Behaviors of problem requirements (*P*-be): as statements about behaviors the design needs to fulfill;
- Features of solutions (*S*-fea): as statements about features of the solutions for the design;
- Behaviors of solution (*S*-be): as statements about how the solutions behave.

4.2

Results

4.2.1

The separation between problem requirements and solutions

The results of the first level of coding are plotted in Fig. 6, in which the horizontal axis presents the time, the vertical axis presents the type of the segment, and discontinuities represent the segments that could not be categorized into either problem specifications or solutions. This diagram clearly substantiates the viability of separating the design process into problem specifications and solutions. The horizontal lines indicate a period of time in which the designer was thinking about either the problem requirements or design solutions, and the vertical lines indicate a shift from one to the other.

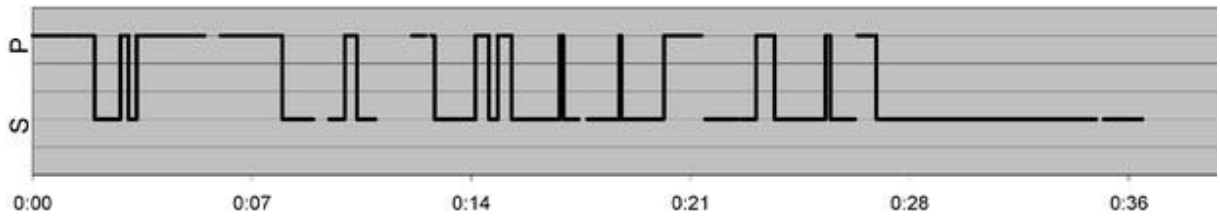

Fig. 6. Representation of the human design process as two search spaces

4.2.2

The qualitative interactions between problem requirements and solutions

To examine the relationship between the segments regarding problem requirements - and those for solutions - we consider the times indicated by a vertical line. These are the periods when the two spaces interact with each other. Computationally, the interactions between them drive the co-evolutionary process forward and make the problem specifications and solution spaces evolve in response to the other. Many interactions are found in this data, and some of the segments are extracted to demonstrate the interactions.

A. After 10 minutes in the design process, the designer examined the existing design and said:

> *Let's see . . . Um, seems like a thick plastic station, It's, (it's, it's) compact and clean but not as (not as) elegant as the things that I would like.*

In this instance, the examination of the current solution stimulates a change in the direction of the behaviors of problem requirements, as illustrated in Fig. 7. He realized his initial thoughts were different from the real product and decided to make a better design. After this, the elegant form became a more essential issue in the design solution. As a result, the direction of problem requirements changes because of the solutions, given that the designer's examination is regarded as one type of reasoning about the solution space.

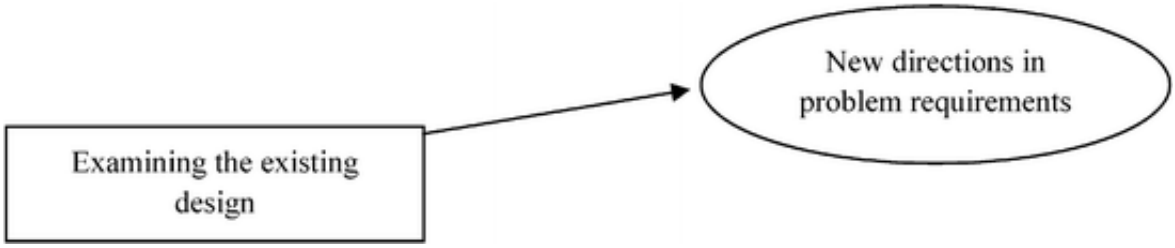

Fig. 7. Examining the existing drawing produces new directions in problem requirements

B. In the middle of the design process when the designer sketched the design solution, he said,

> *cause it (needs, be) needs still, be sort of recognizable as the same as the same product (sketching). So I guess obviously needs a logo somewhere, (and) a band there've got here, and . . .*

The designer was trying to modify the shape of the lid in one of the current solutions and realized the logo and a band were needed. The former, a logo, is a new problem requirement, that is added into consideration, while the latter, a band, is an old problem requirement. In this case, the solution affects the generation of both new and old problem requirements, as illustrated in Fig. 8. It is the interaction between problem requirements and solutions in the co-evolutionary model in which the problem requirements are evolved.

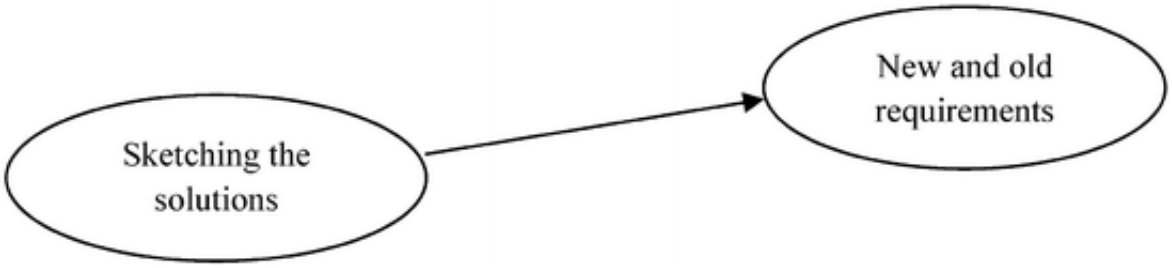

Fig. 8. Sketching triggers new and old problem requirements

In addition, this is a good instance to show the role of drawings as vehicles for thinking about solutions. The designer tried to list all the possible functions in the beginning. However, there were always some additions or modifications of the problem requirements after sketching the solutions. The sketches provide the means to test the solutions in mind or on the papers. It becomes realistic to regard the design process as co-evolution with the evolution and interaction between sketching/examining and problem requirements of the artifacts to be designed.

C. In the middle of the design process, while reasoning about the problem requirements, the designer reported,

> *Um . . . but the lid . . . the handle is my major concern. Maybe I need try (to) go back that and try (and) integrate . . .*

In this segment, the reasoning about problem requirements drives the designer's sketching and thinking about solutions from sketches. He recognized that the lid was an important issue, so he then tried to put his thoughts about this into sketches, as illustrated in Fig. 9. This model is frequently found in the whole design process. The sketches are related to not only the lid but also to the whole area or even to the whole design. This means drawing is not only putting the thoughts down, but it also facilitates thinking about the solutions partially or wholly.

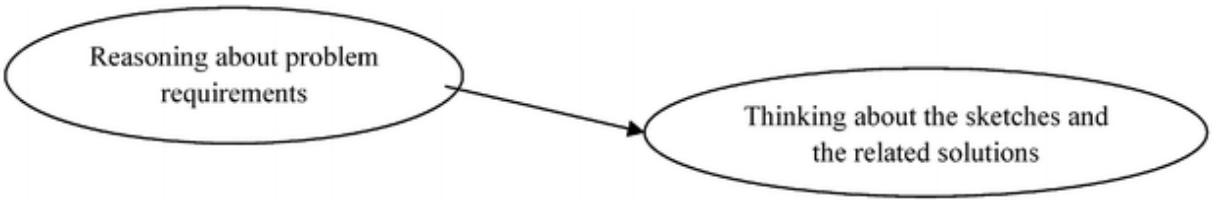

Fig. 9. Reasoning in the problem specifications leads to sketching

D. In the design process the designer thought about the lid. During that period he reported,

> *but the thing I reckon the major issues probably the lid maybe needs a little bit more of a lip kind of thing . . . 'cause it (needs, be) needs still, be sort of recognizable as the same as the same product. So I guess obviously needs a logo somewhere, a band there've got here . . . Makes the kettle look a bit more . . . um . . . a bit more technical. I guess at least it looks better. Um . . . but the lid . . . the handle is my major concern . . . Maybe I need, try, go back, that and try, integrate . . . They want this band on there . . .* You need straight, upright flat shape of the kettle.

The essential requirement, lid, caused the oscillation between problem requirements and solutions. It becomes the operator between a series of problem requirements and solutions. The result of the problem requirements is put into the solutions, and then the thinking of the solutions trigger the reasoning about problem requirements. The process iterates until satisfactory solutions appear, as illustrated in Fig. 10. Although in our data these sentences are not consecutive, thinking about the lid plays an important role in associating different solutions and problem specifications.

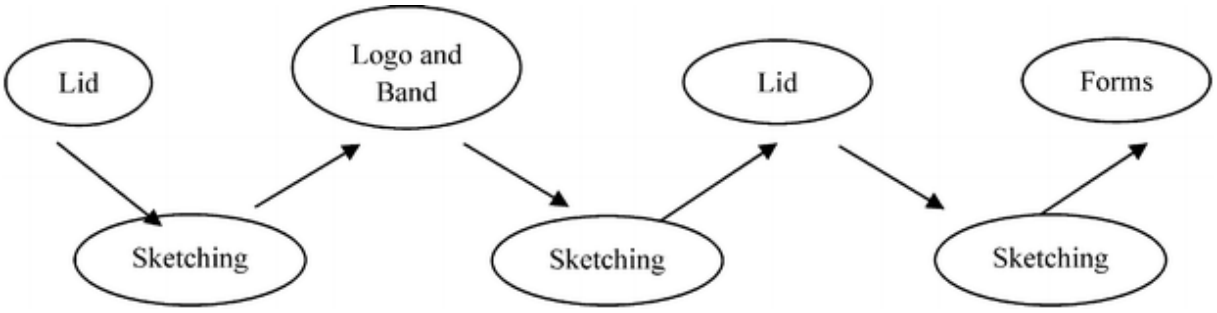

Fig. 10. Interaction between two spaces

4.2.3

The evolution of the problem requirements

One of the features in the co-evolutionary model is the evolution of the problem requirements. In design, the problem requirements are not complete or explicit when the process commences. While some problem requirements are stated initially, others come along during the process of creating solutions. These additional problem requirements can be an elaboration of the initial problem requirements into more detailed requirements, or they can be in response to difficulties encountered when developing a particular design solution.

To explore the evolution of problem requirements, we isolate the designer's reference to the problem features *P*-fea (Table 1). The table shows Code, the code assigned to the segment, and Features of the problem requirements that are associated with that code and their associated duration in time. The local counter for time *t* is given as a duration of the cycle in which the problem requirements were being considered. The global counter *T* is given as the duration of a co-evolutionary cycle in which a set of problem requirements and design solutions were being considered. In this data set, we have 14 units of consecutive problem requirements labelled as *P*-fea. We found that the designer does not consider the entire set of problem requirements at one time, as in the computational co-evolutionary model. One part of the problem requirements was examined at one time. Some important problem requirements were considered several times, such as the lid in this data.

Table 1. The evolution of the features of problem specifications

Code	Features of problem specifications	*t*	*T*
P-fea 1	Speedo and ergonomic	0:02:02	0:02:53
P-fea 2	Form and market	0:00:16	0:03:25
P-fea 3	Marker/ readout, temperature, safe lid, large handle, insulated outer, stable, pouring, power	0:02:14	0:06:10
P-fea 4	Button, lid (P3), power (P3), market (P2), compact and clear, switch	0:02:02	0:10:15
P-fea 5	Lid (P4)	0:00:30	0:12:27
P-fea 6	Lid (P5)	0:00:45	0:14:33
P-fea 7	Lid (P6), handle (P3)	0:00:26	0:15:17
P-fea 8	Logo	0:00:27	0:17:18
P-fea 9	Lid (P6)	0:00:08	0:19:16
P-fea 10	Band	0:00:04	0:20:44
P-fea 11	Capacity, cost/market (P4)	0:01:21	0:23:46
P-fea 12	Speed, temperature meter (P3)	0:00:36	0:26:02
P-fea 13	Wide base, cheap (P10)	0:00:10	0:27:05
P-fea 14	Power plug-in (P4)	0:00:38	0:36:27

Furthermore, we qualitatively group the units of problem requirements as they were in one cycle of co-evolution, in which the search of the problem requirements space has converged. In our data, three distinct groups are found, *P*-fea 1~3, *P*-fea 4~9, and *P*-fea 10~14. During the time from *P*-fea 1 to *P*-fea 3, this designer was trying to generate as many functions for the design as possible. Some previous studies call it "the primary generator" or "the organizing principle". Consequently, we can find most of the rest of the problem requirements are related to these problem requirements in three units of the first groups. For example, the temperature meter shown in *P*-fea 12 appeared initially in *P*-fea 3. Among these features of problem requirements, the "lid" appears most frequently. The evolving process could be traced through the series of problem requirements, *P*-fea 3, *P*-fea 4, *P*-fea 5, *P*-fea 6, and *P*-fea 9. Possibly because of the limited duration, most of the *P*-feas appear twice. Given the observation of a long-term design process, it may be possible to find more changes of problem requirements over time.

4.2.4

The transitions resulting in changes of problem requirements

The interaction between problem requirements and solutions is another important feature of the co-evolutionary model. In Sect. 4.2.1, it was shown that the design process could be regarded as two interacting spaces, while in the following two sections, various types of transitions are illustrated in detail. We first examine the transitions resulting in changes of problem requirements in the design

process, and then the transitions resulting in changes of solutions.

Theoretically, three different types of transitions in changes of problem requirements could be found, due to the combinations of the four elements of the coding scheme. However, types of transitions are richer in terms of the content, which varies the combinations between problem requirements and solutions. Consequently, we further divide these four categories into subcategories in order to highlight more detail.

Four different types of changes of problem requirements are found from the analysis and observation of data.

- Adding new problem requirements (ADD): a design problem is now well known as an ill-defined problem, without clear problem requirements in the beginning, so designers generally find new problem requirements during the design process. This kind of change in problem requirements is named "adding new problem requirements", which extends the boundary of the space of problem requirements.

- Refining problem requirements (REF): the initial problem requirements are often modified according to how designers apprehend and reframe the design problem. This kind of change is named "refinement of problem requirements", which refines the information within the original bounds of the space of problem requirements.

The above two types of changes modify the space of problem requirements; however, we enlist two other thinking processes, which do not modify the problem requirements, but may lead to changes of the problem requirements.

- Searching for new problem requirements (SEA): the effort of designers in finding new problem requirements is not always conclusive, but is a search process without immediate results. When this occurs, the corresponding protocol is encoded as a "search for new problem requirements".

- Re-examination of problem requirements (REE): the initial problem requirements generally must be fulfilled, no matter how the design requirements are interpreted during the design process. Consequently, designers often re-examine the initial problem requirements to assure their fulfillment. These periods are regarded as "re-examination of problem requirements".

The following group of segments illustrates how the design requirements were changed as a result of reasoning about the current best problem requirements and solutions, using the classification above.

a. ADD. Reasoning about the behaviors of problem requirements results in adding new problem requirements, for example, the transition in C8 (*P*-be $\rightarrow$ *P*-fea).

'Cause it ,needs, be, needs still (to) be sort of recognizable as the same as the same product.

So I guess obviously needs a logo somewhere, a band there've got here, and . . .

b. REF. Reasoning about the behaviors of problem requirements results in refinement of the problem requirements, for example, the transition in C7 (*P*-be $\rightarrow$ *P*-fea).

Even, make it a bit more interesting

But the thing I reckon the major issues probably the lid maybe needs a little bit more of a lip kind of thing

c. ADD. Thinking about the solutions results in adding new problem requirements, for example, the instance in C3 (*S*-fea → *P*-fea).

So I have, think of some kettle designs I will take them all . . . , let's see . . . [were thinking]

It would be nice if people knew exact how the hot water was just so they knew (m: how, pour the water)

d. ADD. Drawings of solutions result in adding new problem requirements, for example, C10, R12 (*S*-fea → *P*-fea).

Maybe I need, try, go back, that and try, integrate . . .

They want this band on there . . .

e. ADD. Examining the solutions results in adding new problem requirements, for example, the instance in R10, R13 (*S*-fea → *P*-fea).

So I just thinking . . . oh . . . this kettle I've drawn was already too small.

If (if) the size of the kettle is a major setting point then my kettle also to be able to be same size.

f. ADD. Reasoning about the interactions between the existing depicted ideas and its environment results in adding new problem requirements, for example, the instance of C15, C16 (*S*-be → *P*-fea).

Tall and thin was prone, tip over by mistake

So this I want a pretty wide base for everything. flare up the base a bit.

g. ADD. Examining the behaviors of the existing depicted ideas results in adding new problem requirements, for example, the instance in C13 (*S*-be → *P*-fea).

Um . . . yeah, I am basically thinking of . . . it looks right now,

I've (I've) decided that . . . scalding-wise

The following group of segments illustrates how the designer was reasoning about the design specifications although changes were not made explicitly, using the classification above.

a. SEA. Reasoning about the behaviors of problem requirements results in searching for new problem requirements, for example, the transition in R6 (*P*-be → *P*-fea).

And the water drips everywhere and I'll see exactly what the problem is, but, no, it (the kettle) was all fine.

So I'm just looking at the overall shape again, trying to see what else could possibly be wrong.

b. REE. Reasoning about the behaviors in the problem requirements results in re-examining the initial problem requirements, for example, the transition in C6 (*P*-be → *P*-fea).

Just trying (to) think what people are getting, . . . um . . . some things like scalds caused by kettles.

Several ways . . . find and modify the interface of the kettle areas that are . . . some expensive electrical devices . . . [reading the brief]

c. REE. Thinking about solutions results in re-examination of the initial problem requirements, for example, the instance in R1 (*S*-fea → *P*-fea).

So my first of the Speedo kettle were just like a streamlined kind of design, streamlined I thought of because Speedo costumes, swimming, streamlined.

So again I was reading through making sure exactly what my requirements are for the project.

d. SEA. Thinking about solutions results in searching for new problem requirements, for example, the instance in R9 (*S*-fea → *P*-fea).

but its probably, I think probably incorporate some features from this design into a kettle that looks more like the other kettle.

Ah, now I was looking at the box and seeing what the actual features of this kettle need to be, what that marketing and setting points are, a matter of meeting the need.

e. SEA. Drawing solutions results in searching for new problem requirements via examining the existing product, for example, C11 (*S*-fea → *P*-fea).

You need straight, upright flat shape of the kettle.

A couple of capacity or compact [looking at catalogue]

f. REE. Drawings of solutions results in re-examining initial problem requirements, for example, R2 (*S*-fea → *P*-fea).

So I was going over that, looking at that, seeing how I could possibly improve the spout design just from my rough drawing,

Umm, still thinking about the main things

g. SEA. Examining the solutions results in searching for new problem requirements via examining the existing product, for example, the instance in R11 (*S*-fea → *P*-fea).

So even though I (sort of) like that shape, it's not quite appropriate.

Looking again at the box, thinking of what it says, 'cause if I was trying to design for this company I have to know what that company wants.

h. REE. Reasoning about the interactions between the existing depicted ideas and their environment results in re-examination of the initial problem requirements, for example, the instance of C2 (*S*-be → *P*-fea).

so that (so that) they (don't doesn't have that) can't spill out here and then,

Um, ok some elegant and beautiful forms

i. REE. Examining the behaviors of the existing depicted ideas results in re-examination of the initial problem requirements, for example, the instance in C9 (*S*-be → *P*-fea).

Makes the kettle look a bit more . . . um . . . a bit more technical. I guess, at least it looks better

Um . . . but the lid . . . the handle is my major concern.

Different transitions resulting in the changes of problem requirements are illustrated above and tabulated in Table 2. In this table, the "Code" is the type of code applied to the segment that initiated the transition to problem requirements, "Type of reasoning" describes what the designer was reasoning about, and "Type of change to problem requirements" refers to the categories described above. The mark "X" indicates that we found this type of change in a transition between vertical and horizontal elements in this data. However, not all types of changes are found for all types of transitions in our data, probably because we had limited data analyzed, or these types actually do not exist in the design process. Therefore, it necessitates the further analysis of more data to examine the transitions completely.

Table 2. The transition types resulting in changes to requirements found in this data set

Code	Type of reasoning	Type of change to problem requirements			
		ADD	REF	REE	SEA
P-be	Reasoning about the behaviors of problem requirements	X	X	X	X
S-fea	Thoughts of solutions	X	-	X	X
	Drawings of solutions	X	-	X	X
	Examining the solutions	X	-	-	X
S-be	Reasoning the interactions between the existing depicted ideas and its environment	X	-	X	-
	Examining the behaviors of the existed depicted ideas	X	X	X	-

4.2.5

The transitions resulting in changes of solutions

This section examines the transitions resulting in the changes of solutions according to the same method applied in the previous section. The salient difference of the analysis in this section from the previous is that the results rely heavily on the observations of the drawing activity of the designer. The relationship between new and old events, either thoughts or drawings, is too vague to distinguish due to the mixture of these events. There are not enough cues to trace each drawing back to its precedent. Moreover, we studied only the verbal protocol in this experiment so that the instances of drawing activities are based on the verbal utterance. Therefore, not all of the drawing activities are reported by the protocol, and this situation increases the difficulty in distinguishing one instance as adding new solutions or as a re-examination of current solutions.

In the computational co-evolutionary model, the solutions space was changed through the application of genetic operators, while in the human design process, it is impossible to see the underlying mechanisms that lead to the changes in the solution. We can only see the changes revealed by superficial verbal description. By protocol and observations, three essential types of changes of the solutions found in the analysis of data are: 1) reasoning about the interactions between the current solution and its environment, 2) examining the behaviors of the current solution, and 3) the intention of the designer to introduce a new concept.

There are three types of changes in features of solutions:

- Thoughts of solutions (THO): a brilliant solution for a design problem may be the concept that breaks the traditional way of solving a problem.
- Drawings of solutions (DRA): the material a designer uses to communicate with peers and clients is drawings. Creating and examining those drawings indicates reasoning about the solution.
- Examining the solutions (EXA): the evaluation of an existing solution is part of the solution since it sets up the priority for the solutions.

All the transitions found in this data result in changes in the space of solutions either by conceptual thought or by physical drawings. There is no transition resulting in the examination of solutions in this data set.

Here we look at how the designer examined the current solution in order to make changes to the design. This is the horizontal evolution of the solution space in the co-evolutionary model of design.

a. DRA. The intention of the designer results in the drawings of features of solutions, for example, the instance in C5.

> *I don't want (to) change it so much so people would be able (to) recognize it. I guess,*
>
> *Um . . . drawing out. I guess what I am drawing out what . . . I am thinking things that I like and don't like.*

b. THO. Examining the behaviors of the current design solution results in the thoughts of features of solutions, for example, the instance in C6.

> *Um, I just find the overall shape little bit boring, sort of, like a Tupperware kind of things. (It) doesn't look like a nice (nice) kettle*
>
> *Maybe (it) need some colors or something.*

c. THO. Reasoning about the interactions between the current design solution and its environment results in the thoughts of features of solutions, for example, the instance in C2.

> *You can push out with your thumb, but flips up the lid, so they don't actually have (to) take the lid off. They can be holding the handle, while they pour it in.*
>
> *I think, (to) pour it you must have good lip at the front.*

d. THO. Reasoning about the interactions between the problematic kettle and its environment results in the thoughts of features of solutions, for example, the instance in C9.

It's half of the water at all time before switch on [read the text on the bottom of the problematic kettle].

Just want to know it's on the top of the market. It might be good (to) have a little (floating bubble to show the height of the water).

e. DRA. Examining the behaviors of the current design solution results in drawings of features of solutions, for example, the instance in C10.

Um this idea works well. Maybe we should trying (to) work in.

Round it a bit more. It's a bit square.

Here we look at the transitions from behaviors of problem requirements to features of solutions. This is the downward arrow in the model of co-evolutionary design.

a. THO. Reasoning about the behaviors of problem requirements via examining the current solution results in the thoughts of features of solutions, for example, the instance in R7.

Just looking at the handle thinking about how it all works.

Yeah, so now I'm thinking, I'd put in the meter thing for . . . that was like the measure, oh.

b. DRA. Reasoning about the behaviors of problem requirements via examining the current solution results in the drawings of features of solutions, for example, the instance in R25.

Looking inside to see how the element works, and see how if I can see any injected points or injection points just to see how its melded.

And I am putting it back there, now to try and draw it again.

c. THO. Reasoning about the behaviors of problem requirements results in the thoughts of features of solutions, for example, the instance in R3.

I think of people pouring hot water and how they go about it so, just what they do.

So I can think of how my kettle design should reflect how people pour water from their kettle umm.

d. DRA. Reasoning about the behaviors of problem requirements results in the drawings of features of solutions, for example, the instance in R27.

On the side and again I was trying to look at how, trying to work out how it's made

First comparing my design to that kettle, trying to see how if there is anything I can do, to sort of, make them work together better.

The transitions that result in changing the design solutions are tabulated in Table 3. Our results show that our designer's transitions between behaviors of solutions and features of solutions occupy 70% of all the transitions resulting in changes of solutions. This phenomenon indicates that reasoning about solutions plays an important role in generating solutions, while the interactions between the problem requirements and solutions are much less frequent. Similar to the computational co-evolutionary model, in each evolutionary cycle the best result is produced after a number of evolutions and

evaluations, and then the result is passed to change the problem requirements.

Table 3. The types of the transitions resulting in the changes of solutions

Code	Type of reasoning	Type of change		
		THO	DRA	EXA
P-be	Reasoning about the behaviors of problem requirements	X	X	-
	Reasoning about the behaviors of problem requirements via examining the current solution results in the drawings of solutions	X	X	-
S-be	Reasoning about the interactions between the current solution and its environment	X	-	-
	Examining the behaviors of the current solution	X	X	-
	The intention of the design	X	-	-
P-fe	Adding new features	X	X	-
	Refinements	-	-	-
	Re-examination	-	-	-
	Searching	-	-	-

4.3 Discussion

Reasoning and examination are two main events found in our analysis of the interactions, excluding free-hand sketching. In pursuing co-evolutionary design as a cognitive model of the design process, we can draw a loose equivalence between examining and reasoning as observed in human designers with the fitness functions and genetic operators of the computational model. Both human reasoning and genetic operators result in the generation of new alternative problem specifications and solutions, while both examination and fitness functions identify the best in one generation of problem specifications and solutions.

In this analysis, we observed a co-evolutionary process between problem requirements and solutions in the human design process. We identified the following four mechanisms related to the evolution of problem requirements:

- adding new problem requirements;
- refining problem requirements;
- searching for new problem requirements;
- 4e-examination of problem requirements .

We identified three mechanisms related to the evolution of the features of design solutions:

- thoughts of new solutions;

- drawings of solutions;
- examining the solutions.

5

Architectural design experiment

5.1 Experimental setting

The protocol data for the architectural design experiment was taken from previous studies (Suwa et al. *1998, 2000*; Tang and Gero *2001a*). In this experiment, voluntary participants were practicing architects with 30 years' experience in residential house design. The device and its arrangement were the same as those in the kettle design experiment, and the experiments were held at the same studios at the University of Sydney.

The final version of instructions for this experiment included warm-up exercises, design, retrospective report, and a brief interview. The separation between design and report made these retrospective protocol studies. Following the instructions, subjects were asked to design without thinking aloud, and then to retrospectively report all details of what they thought while solving the problem. To aid in the quality of the report, the videotapes recording the design session were used for visual cues. The experimenter did not interfere with the subjects, and the subjects were asked to report what they thought instead of explaining their design concepts to the experimenter. The experimenter sat next to the subjects and kept tracing the reporting progress to ensure that the subjects reported relevant information about the design.

5.2

The design brief

The design brief for this experiment was to design a house for a couple. The subjects were given the following information. The female, age 29, is a dancer, and the male, age 34, is a painter. They are sensitive to color and beauty and enjoy contact with the natural environment. The site is located on the corner of fully serviced home sites surrounded by a large central open-space recreation reserve in Matraville, one of Sydney's southeastern newly desirable locations. It is trapezoidally shaped and slopes down to the edge of the recreation reserve. The site has a view of the flame trees in the recreation reserve and the whole reserve. The site area is 700 m^2. The floor-space ratio for this site is 0.65:1, so the maximum floor plan can be 455 m^2. The house will have gentle sea breezes and is screened by a stately grove of flame trees along the edge of the estate. A sculpture garden is required to display their art collections. According to the Randwick Development Control Plan No. 4, the height of a dwelling house should not exceed a maximum of 9.5 m. The design task was to give form to and to arrange the spaces on the site with the approximate sizes shown in Table 4.

Table 4. Design brief data

Space	Area (m^2)
Living/dining area	40
Kitchen	15
Bath	10
Master bedroom	30
Bedroom	20
Painter’s studio	50
Dancer’s studio	50
Observatory	20
WC-shower	9
Parking space	36

5.2.1 Segmentation

Segmentation for this study is important because it determines the results of the transitions between consecutive segments that are essential for observing the co-evolutionary process. Segmentation enables the clear observation of transitions between problem and solution spaces. In the previous experiment, we divided the design protocol along lines of designers’ intentions and actions. In this way, the content of segments is closer to that of the smallest units of the thinking process. Therefore, the designer’s intention is interpreted, and each segment represents a single intention of the designer.

Similarly, we applied the same method in this experiment, but further parsed the segments into smaller units, based on the notion of "discussed issues". They are considered as nouns of the utterances in each segment. To illustrate, the same protocol is parsed and encoded by both of the methods used in the kettle design and the architectural design experiments. In the tables below there are two columns representing the different levels of the coding scheme, which will be described in the next section. In the first column, the segment is coded as Problem or Solution and in the second column the segment is coded as Function (F), Behavior (B), or Structure (S) and we use SI if the issue is the design site. The segmentation in Table 5 is based on the designer’s intention only, and the segmentation in Table 6 is based on the intentions and nouns in the same protocol. The encoded results show that the latter method gives us more detailed transitions in the design process, and preserves the essential encoding, such as the P in P/S spaces and the S in the F/B/S code. We have more information when parsing the protocol based on both intentions and discussed issues, and it helps us identify the co-evolutionary issues that designers deal with.

Table 5. The encoded segment based on intentions

Segment	P/S	F/B/S
The features of the site where the views might be, the orientation of the sun, and to get the feel of the actual area of the site	P	SI

Table 6. The encoded segment based on intentions and nouns

Segments	P/S	F/B/S
The features of the site	P	F
Location of the views	P	B
Orientation of the sun	P	SI
To get the feel of the actual area of the site	P	B

For another example, the segmentation in Table 7 is based on intentions. In Table 8, three more discussed issues were identified. The parsed results demonstrate a design situation that a designer tends to think about related issues together in an intention. This means the designers thought about the three spaces in the house and intentions. However, at the same time four issues were discussed: the dance studio, the painting studio, the observatory room, and the collection of these three. The separated nouns enrich the transitions that we can observe, and now we can observe not only transitions between intentions but also between discussed issues.

Table 7. The encoded protocol based on intentions

Segment	P/S	F/B/S
Unlike most houses that I had, three spaces that I didn’t really understand: the dance studio, the painting studio, and the observatory room	P	F

Table 8. The encoded protocol based on nouns

Segments	P or S	F/B/S
Unlike most houses that I had, three spaces	P	F
That I didn’t really understand: the dance studio	P	F
The painting studio	P	F
The observatory room	P	F

In parsing discussed issues, it should be noticed that we considered the shared context in the same segment to distinguish the different discussed issues. For example, the three discussed issues in the F/B/S column of Table 9 were considered as coherent issues, so all of them were encoded as behaviors. However, if we considered the issues individually, the first two issues would be different. "The living areas" and "dining and living", respectively, would be coded as Structures according to the definitions.

Table 9. The example of segmentation and encoding

Transcript	P/S	F/B/S
So the living areas / dining and living / would actually be orientated to the north-east and look out to the best outlook	S/S/S	B/B/B

This is similar to encoding a segment. Sometimes it has to be based on other segments that have an intentional relationship with it, and they could be the following or previous segments. For example, the segment in Table 10 is hard to define by itself, and consequently, the following segment is considered

in order to understand the designer's intention to encode the content.

Table 10. The example of encoding a segment

Transcript	P/S	F/B/S
But I am still just looking at	P	S

5.2.2

Coding scheme

For this design experiment, the first level of coding is the two co-evolutionary spaces, where problem requirements (*P*) and solutions (*S*) spaces are the essential categories for investigating the diagonal behaviors of co-evolution. To clarify, the definition of problem requirements space is the articulations of requirements, specs, needs, and desires that a designer wants in the design solution. For example, the first segment in Table 11 is about a confusing issue found in the structure, and it is encoded as solutions space. The following segments are encoded as in the problem spaces, since the contents are about the design requirements that the designer wanted to achieve.

Table 11. The example of encoding a segment

Number	Transcript	P/S
1	And then to the bedroom, the main bedroom was a bit of a problem because initially I sketched it, ah, a bit more square	S
2	And I wanted to orient it to the courtyard	P
3	I wanted to make the courtyard as big as possible.	P
4	So that it wasn't crammed against the studio at the back	P

The second level of coding is based on the prototype framework: function, behavior, and structure (Gero and Rosenman *1990*), which are applied to describe the content of the problem specifications and solutions spaces. In Gero and Rosenman's definition:

- function (F) relates to the purpose of an artefact, and when in the Problem space it refers to the intended or desired purpose, when in the Solution space it refers to the actual purpose;
- behavior (B) relates to the actions or processes of an object or artefact, again in the Problem space it is the expected behaviors and in the Solution space it refers to the actual behaviors; and
- structure (S) relates to the definition of objects and their relationships that comprise a physical solution, where in the Problem space it refers to required structure and in the Solution space to actual structure.

In addition to these three, two new subcategories are added to account for the context of the design of a residential house.

- site specification (SI) related to the original requirements for the building site.

- site structure (ST) related to the additional requirements for the building set up by a designer.

Based on the context of design task and the experience from encoding data, we extend the working definitions of function-behavior-structure as follows.

Function includes the purpose of the rooms and the activities intended to occur in the space, for example, "*because the activities would tend to be more internal rather than outward looking*" is encoded as function. Function can apply to segments that are encoded as a Problem requirements or Solutions space. For example, "*that we're going to need to be put into the house*" is encoded as a problem space/function segment, and "*that we're going to put into the house*" is encoded as a solution space/function segment.

Behavior includes the performance measure of the design, and views, for example, "*so going up to a second story was not going to benefit the living in the house*". This segment is encoded as behavior. Behavior can also be applicable in both problem specifications and solutions spaces. For example, "*to get the feel of the actual area of the site*", is encoded as a problem space/behavior segment, and "*and having roughly divided them into three groups*" is encoded as a solutions space/behavior segment. Moreover, the segments including the designers' evaluation are encoded as behavior, so this kind of segment may be related to the following vocabularies: number, fit, close, enough, bigger, smaller, and work.

Structure includes the locations and adjacencies of spaces, for example, "*so I assume the council normally have a set back building lines set back*" is encoded as structure. An example of a problem space/structure segment is "*I've started off with the simplest thing, the garage*". An example of solution space/structure segment is "*So I set out the actual area that I thought would probably be the (the) allowable building area*".

This coding scheme provides a basis for 1) examining the evolutionary processes of problem specifications and solutions spaces, 2) examining the content of problem requirements and solutions spaces, and 3) closer examination of transitions since (smaller) nounal segments were coded instead of (larger) intentional segments.

5.3

Results

In this section, we describe the statistical results of the distribution of occurrences in problem requirements and solutions spaces, and the distribution of occurrences in function/behavior/structure spaces.

5.3.1 The distribution of occurrences in problem space and solution space

The distribution of occurrences in problem specifications and solutions spaces in our analytical data is plotted in Fig. 11. Each dot in the figure presents a segment that we regarded as in either problem specifications or solutions spaces. The segments that are indistinguishable in terms of problem/solution spaces are excluded from this figure.

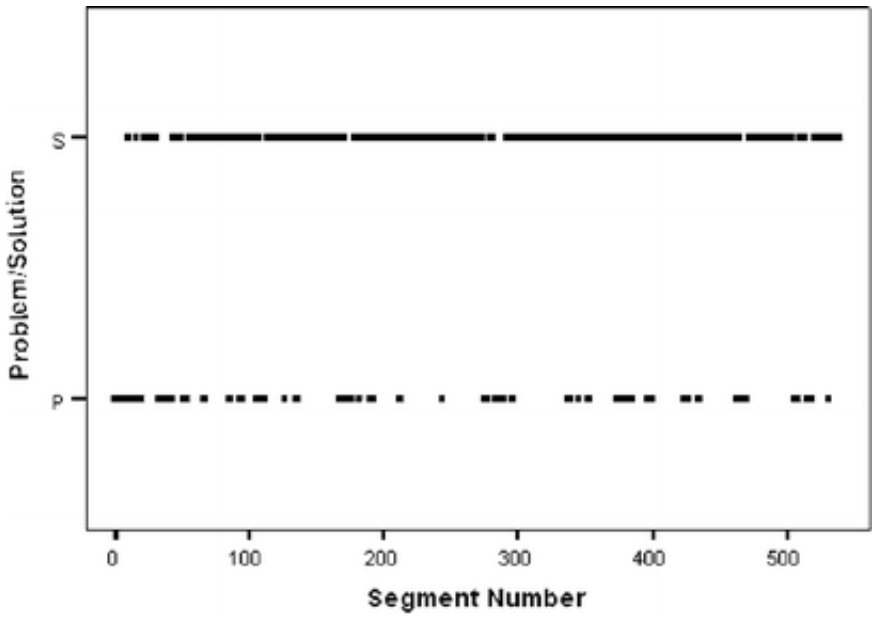

Fig. 11. The occurrence of problem and solution spaces: *P* indicates the thinking of problem and *S* indicates the thinking of solution spaces

---

Figure 11 demonstrates two distinguishing features. First is the co-evolution of problem and solution issues in the human-thinking process. Despite the difference in the number of segments between the two spaces, the thinking of both problems specifications and solutions occurred pervasively in the whole design process. The observed frequency of the reasoning about requirements does not decrease dramatically in the end of this conceptual design. What we can propose here is that the conceptual designing in this case study could be regarded as a co-evolutionary process between problem requirements and design solutions spaces.

Second, we observed an unbalanced amount of reasoning in the problem and solution spaces. The left pie chart in Fig. 12 shows the amount of segments that relate to problem requirements space is two quarters less than the amount for the solutions spaces. The trend shows that this designer spent more time thinking about the solutions in the design process. This is different from the computational co-evolutionary model in which we tend to give the same number of evolutionary circles to both spaces, shown in the right pie chart, Fig. 12. It may be interesting to apply this feature to the computational co-evolutionary model to examine whether this unbalanced amount of reasoning is beneficial to the computational performance.

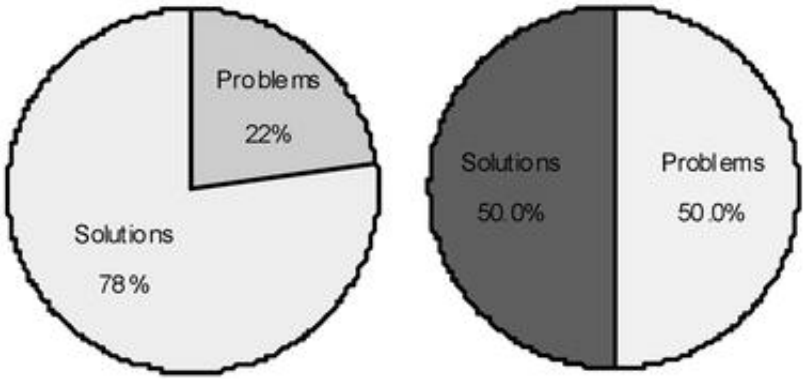

Fig. 12. The relative amounts of time spent for problem and solution space in the human *left*, and computational design process *right*

To further examine the trend of reasoning between problem requirements and solution spaces, we plotted the occurrences of segments in both spaces in percentile comparison (Fig. 13). A comparison percentile plot allows us to compare the distributions of segments in two spaces. The points plotted in the graph in Fig. 13 are the 1, 2, 3, 4, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 95, 96, 97, 98, and 99th percentiles. The 50th percentile point indicates that the 50th occurrence of reasoning in the solution space was in about segment 300, and the 50th occurrence of reasoning in problem space was in about segment 200. This indicates that the segments of reasoning in the requirements space appear earlier than those in the solutions space. It means that the designer reasoned in problem space more in the first half of the design process. This phenomenon is different from the reasoning process in the computational model. It would also be interesting to apply this phenomenon to the structure of computational models, changing the time distribution in reasoning different spaces.

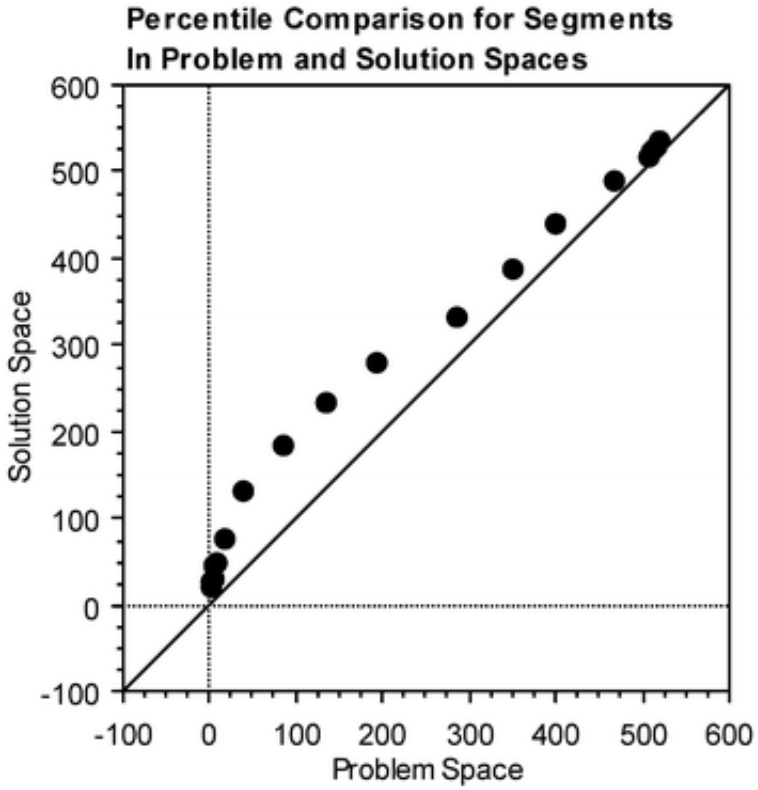

Fig. 13. Percentile comparison for segments in problem and solution spaces

In this experiment, a single segment was determined to be a single intention of the designer, rather than the features of the requirements/solutions as discussed before. Therefore, an episode of reasoning in the requirements space might span through several segments. To demonstrate this situation, we calculated the average continuous segment length in the requirements space and in the solution space. The results are 3.3 segments and 11.3 segments, respectively. This shows that a designer reasoning in the solution space expresses more intentions continuously before reconsidering the requirements space.

5.3.2

The transitions between function, behavior, and structure in terms of problem and solution spaces

To observe the transitions between function, behavior, and structure, we plotted the instances of thinking about them, and further divided them in terms of problem requirements and solutions spaces (Fig. 14). The problem space/structure segments are divided into two subcategories, site specification (SI) and site structure (ST), which were presented in Sect. 5.1.3.

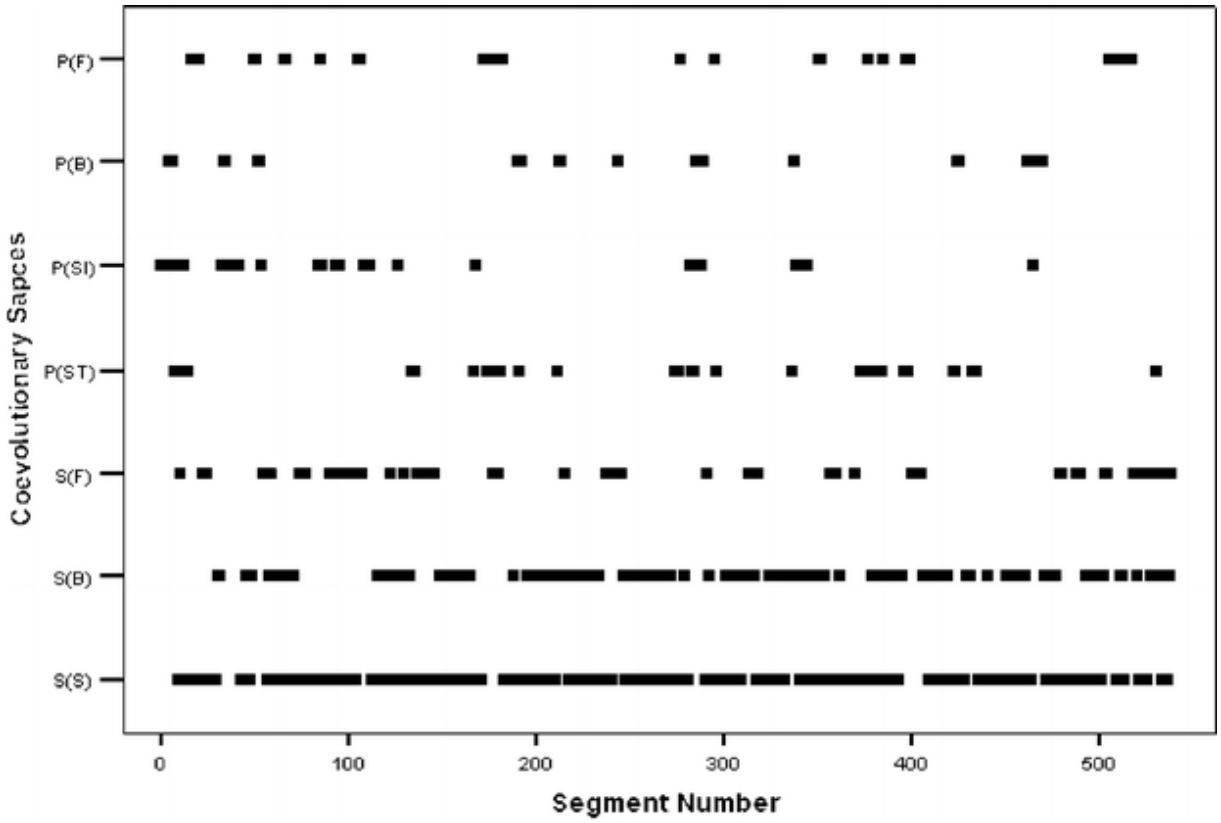

Fig. 14. The transitions between function, behavior, and structure: *P*(F) indicates the problem space/function segment, *P*(B) indicates the problem space/behavior one, *P*(SI) indicates the problem space/ site specification one, *P*(ST) indicates the problem space/structure one, *S*(F) indicates the solution space/function one, *S*(B) indicates the solution space/behavior one, and *S*(S) indicates the solution space/structure one

---

Figure 14 shows that different types of thinking occur throughout the design process. Reasoning in function, behavior, site specification, and site structure in the problem spaces still occurred after the 450th segment, and reasoning of function, behavior, and structure in the problem spaces still occurred until the end of the design process.

We calculate the mean, frequency, skewness, and standard deviation of the number of segments in terms of each category, shown in Table 12. The skewness statistic of less than 1 suggests a symmetrical distribution in a category, and most of the categories have symmetrical distributions, except *P*(SI). Although the distribution is even, the observed frequency in each category varies, ranging from 218 to 63. The solutions space/structure segments are the most frequent activity in this case study.

Table 12. The mean and standard deviation of the segment number in different spaces

	S(F)	*S*(B)	*S*(S)	*P*(F)	*P*(B)	*P*(SI)	*P*(ST)	Total
Mean	255.5	292.2	286.3	258.6	265.5	111.0	274.7	270.5
Frequency	63	137	218	32	23	32	34	539
Std. deviation	171.8	134.3	151.1	192.8	172.4	120.2	143.0	156.2
Skewness	.344	-.026	-.088	.127	-.209	1.497	-.434	-.001

The numbers of structure type segments in both problem requirements and solution spaces exceed that in other types. To verify this, a chi-square test for the distributions of function, behavior, and structure in the two spaces was conducted, (Table 13). To validate this test, site specifications and site structure are merged into structure in the problem space.

Table 13. The count of segments in different spaces

	Function	Behavior	Structure
Problem space	**32**	23	**66**
Solution space	63	**137**	216
Chi-square test result	$\chi^2=13.05 > \chi^2(2)=9.21$ (p-value<0.01)		

The result indicates that there is a statistical significance in the difference between problem requirements and solutions spaces in terms of function, behavior, and structure. The segments that enable the distance between these two spaces are the functional and structural concerns in the problem space and the behavioral concerns in the solution space. The bold numbers in two spaces represent the number that exceeds the expected count in the case where all cells are independent.

In this architectural design study, the interesting finding is the difference between the computational co-evolutionary model and the human design process. It is shown in this paper that the human design process could be appropriately presented as a co-evolutionary process, and the statistics demonstrate the even distribution in different categories. However, the difference between the numbers of occurrences shows the difference in the mechanism between co-evolutionary computational and cognitive models of design.

6 Conclusions

The two protocol studies clearly show evidence of the transitions in the computational co-evolutionary model: *P* to *P*, *P* to *S*, *S* to *P*, and *S* to *S*. The protocol studies also identify the nature of these transitions and suggest the transitions are both temporal and have some basis in the content of the design information. This validates our hypothesis that co-evolutionary design can be the basis for a cognitive model of design. In computational co-evolution, selection, crossover, and mutation are genetic operators that we have used to facilitate the generation of the best problem requirements in one evolution cycle. We are not able to observe the mechanisms used by human designers to compare directly with computational co-evolutionary design, but the analytical data demonstrates that the characteristics of both human cognition and computational algorithms are similar in terms of co-evolutionary cycles.

For human designers, this process is more like a depth-first search due to the powerful reasoning and limited cognitive abilities of human being. In contrast, for computers, it is more like a breadth-first search because of their relatively large memory and limited abilities in reasoning. As a result, the use of selection, crossover, and mutation give computational designing a relatively simple algorithm, compared to human reasoning. This indicates that the computational model of co-evolutionary design complements the human designer's co-evolutionary cycles: the computational model can examine and generate large numbers of designs and the human designer reasons about a few designs.

References

Akin Ö (1984) An exploration of the design process. In: N Cross (ed) Developments in design methodology. Wiley, London, pp 189-208

Akin Ö (1993) Architects' reasoning with structures and functions. Environment Planning B 20:273-294

Chan CS (1990) Cognitive processes in architectural design problem solving. Des Studies 11:60-80

Cross N, Christiaans H, Dorst K (eds, 1996) Analysing design activity. Wiley, Chichester

Dorst K (1996) The design problem and its structure. In: Cross N, Christiaans H, Dorst K (eds) Analysing design activity. Wiley, Chichester, pp 17-34

Dorst K, Dijkhuis J (1995) Comparing paradigms for describing design activity. Des Studies 16:261-274

Eastman CM (1970) On the analysis of intuitive design process. In: Moore G (ed) Emerging methods in environmental design and planning. MIT Press, Cambridge, Mass, pp 21-37

Eckersley M (1988) The form of design process: a protocol analysis study. Des Studies 9:86-94

Ericsson KA, Simon HA (1993) Protocol analysis: verbal reports as data. MIT Press, Cambridge, Mass

Gero J, Tang HH (1999) Concurrent and retrospective protocols and computer-aided architectural design. In: Gu J, Wei Z (eds) CAADRIA 99. Shanghai Scientific and Technological Literature House, Shanghai, China, pp 403-410

Gero J, Tang HH (2001) Differences between retrospective and concurrent protocols in revealing the process-oriented aspects of the design process. Des Studies 21:283-295

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

Gero JS, Rosenman MA (1990) A conceptual framework for knowledge-based design research at Sydney University's design computing. Art Int Eng 5:65-77

Goldschmidt G (1994) On visual design thinking: the vis kids of architecture. Des Studies 15:158-174

Guindin R (1990) Knowledge exploited by experts during software system design. Int J Man-Machine Studies 33:279-304

Larkin J, Simon HA (1987) Why a diagram is (sometimes) worth ten thousand words?. Cogn Sci 11:65-99

Maher ML (1994) Creative design using a genetic algorithm. Computing in Civil Engineering. ASCE, pp 2014-2021

Maher ML, Poon J (1996) Modelling design exploration as co-evolution. Microcomp Civil Eng 11:195-210

Maher ML (2001) A model of co-evolutionary design. Eng with Computers 16:195-208

McNeill T, Gero JS, Warren J (1998) Understanding conceptual electronic design using protocol analysis. Res Eng Des 10:129-140

Mitchell M (1996) An introduction to genetic algorithms. MIT Press, Cambridge, Mass

Paredis J (1998) Coevolutionary algorithms. In: Back T, Fogel D, Michalewicz Z (eds) The handbook of evolutionary computation. Oxford University Press, Oxford

Schön DA (1983) The reflective practitioner. Harper Collins, New York

Simon HA (1969) Sciences of the artificial. MIT Press, Cambridge, Mass

Simon HA (1992) Sciences of the artificial, 3rd edn. MIT Press, Cambridge, Mass

Stauffer LA, Ullman DG (1991) Fundamental processes of mechanical designers based on empirical data. J Eng Des 2:113-125

Suwa M, Gero J, Purcell T (1999) Unexpected discoveries: how designers discover hidden features in sketches. In: Gero JS, Tversky B (eds) Visual and spatial reasoning in design. Key Centre of Design Computing and Cognition, University of Sydney, pp 145-162

Suwa M, Gero JS, Purcell T (2000) Unexpected discoveries and S-invention of design requirements: important vehicles for a design process. Des Studies 21:539-567

Suwa M, Purcell T, Gero J (1998) Macroscopic analysis of design processes based on a scheme for coding designers' cognitive actions. Des Studies 19:455-483

Suwa M, Tversky, B (1997) What do architects and students perceive in their design sketches? A protocol analysis. Des Studies 18:385-403

Tang HH, Gero J (2001a) Cognition-based CAAD. In: de Vries B, van Leeuwen J, Achten H (eds) CAADFutures 2001. Kluwer, Dordrecht, pp 523-531

Tang HH, Gero J (2001b) Sketches as affordances of meanings in the design process. In: Gero J, Tversky B, Purcell T (eds) Visual and spatial reasoning in design II. Key Centre of Design Computing and Cognition, University of Sydney, pp 271-282