Using a Generic Method to Study Software Design Cognition

Jeff Wai Tak Kan
Krasnow Institute for Advanced Study
Fairfax, USA
Taylor's University College, Malaysia
Email: kan.jeff@gmail.com

John Gero
Krasnow Institute for Advanced Study
Fairfax, USA
Email: john@johngero.com

Somwrita Sarkar
University of Technology
Sydney, Australia
Email: somwrita@gmail.com

***Abstract*—This paper presents a generic coding scheme utilized in a protocol analysis to study professional software designers. The paper commences by presenting the hypothesis that the fundamental processes of designing are the same irrespective of the domain and of the artifact being designed. The Function-Behavior-Structure (FBS) ontology has been proposed as the basis of a generic protocol analysis coding scheme. We compare the design processes of the beginning, middle and end of the design session. We also compare the beginning design processes with an architectural design session. The results obtained from using this coding scheme show that a generic method allows for the comparison of designing between different design disciplines. It shows there are differences in designing in terms of design processes between the beginning and the end of the design session. The results indicate that the mix of the design processes involved in architectural design and software design are different.**

I. Introduction

Over the past three decades, protocol analysis has become one of the most widely used methods to study human design activities and cognitive design processes [1], [2]. However, unlike many other research domains there is a lack of agreement on both the terminology and the research methodology. In studying designers there is no uniformity in the segmentation and coding of protocols. The papers in DTRS2 [1] and 11 years later in DTRS7 [2] are examples of the diversity of methods and coding schemes despite studying the same raw data. This paper uses a generic coding scheme to study professional software designers. By using this generic method, the results from this study can be used to compare with other studies that use the same generic scheme.

II. FBS Design Ontology

In order to establish a common ground to study design activities, we propose to use an ontology as an overarching principle to guide the protocol study. Gruber [3] defines ontology, apart from its use in philosophy, as an explicit specification of a conceptualization. Knowledge of a domain is represented in a declarative formalism in terms of a set of objects and their relationships. An ontology may be thought of as the framework for the knowledge in a field.

A design ontology is described by defining the representational terms and their relationships. Gero [4] viewed designing, within an issues and information processing model, as a purposeful act with the goal to improve the human condition. Gero [4] stated: "The meta-goal of design is to transform requirements, more generally termed functions which embody the expectations of the purposes of the resulting artefact, into design descriptions. The result of the activity of designing is a design description."

The coding of the protocols will be based on a general design ontology, namely, the Function-Behavior-Structure (FBS) ontology [4], Figure 1 and its extension, the situated Function-Behavior-Structure (sFBS) ontology [5] as a principled coding scheme (as opposed to an ad hoc one). Scholar.google indicates that the papers mentioned above, which outline the ontology, have over 850 citations between them. A Monte Carlo simulation of 50 selections from these citations, excluding self-citation, shows the breadth of the disciplines the citations come from.

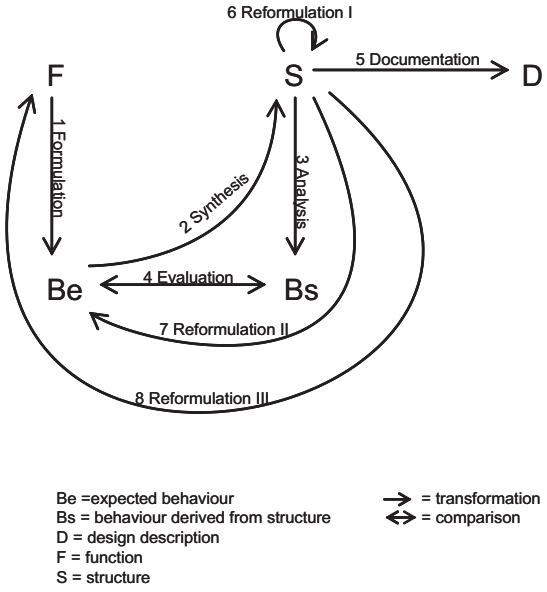

Fig. 1. The FBS ontology of design processes

The FBS design ontology [4], as a formal model, models designing in terms of three fundamental, non-overlapping, classes of issues (variables): function (F), behavior (B), and structure (S); along with two external classes: design descriptions and requirements. In this view the goal of designing is to

transform a set of functions into a set of design descriptions. The function of a designed object is defined as its teleology; the behavior of that object is either expected (Be) or derived (Bs) from the structure (S) that is the components of an object and their relationships. A design description cannot be transformed directly from the functions, which undergo a series of processes among the FBS variables. Figure 1 shows the eight FBS design processes in relation to the function (F), behavior (B), and structure (S) state variables. Formulation (F→Be) is the transformation of the design problem and converts function into expected behavior. Synthesis (Be→S), is the transformation of the expected behavior (Be) into a solution structure. Analysis (S→Bs) is the derivation of "actual" behavior from the synthesized structure (S). Evaluation (Bs↔Be), is the comparison of the actual behavior (Bs) with the expected behavior (Be) to decide whether the solution structure (S) is to be accepted. Documentation (S→D), is the externalization of any design description.

Traditional models of designing iterate the analysis - synthesis - evaluation processes until a satisfactory design is produced. The aim of introducing the three types of reformulations is to expand the design state space so as to capture the innovative and creative aspect of designing. They have not been well articulated in most models because they have not been adequately understood.

Reformulation type I (S→S'), addresses changes in the design state space in terms of structure variables or ranges of values for them. Reformulation type II (S→Be'), addresses changes in design state space in terms of behavior variables. A review of synthesized structure may lead to the addition of expected behavior variables. Reformulation type III (S→F'), addresses changes in design state space in terms of function variables.

A. Software Designing and FBS

The basis of the FBS view is that all design tasks, irrespective of domain, (for example: both architectural design and software design), can be described using a generic FBS framework – as any design task will have functions (F) to fulfill which the object is being designed for, and there will be resulting behaviors (Be and Bs) of any synthesized structure (S). The main advantage of such an approach is that the FBS view can be used to analyze design tasks from various domains under a common ontological structure.

In designing physical objects, the manifestation of structure (S) variables will usually resemble some physical aspect. For example, to design a portable shelter; the function (F) of shelter can be formulated to an expected behaviors (Be) of a space with simple erection method that provides protection. This may be synthesized into an "A-frame" structure (S). With this structure, one can analysis its behavior (Bs), for example headroom and structural stability. After evaluating the behaviors, the designer may accept or reject this structure (S).

However, in the design of software the structure (S) will not have any physicality. In a typical object-oriented program, the objects are referred to as structure. The programmers (designers) formulate the expected behaviors from the functions of these objects. With these expected behaviors they can synthesize the codes or the relationships of codes of those objects. With these objects they can derive their behaviors by either running that part of the program or mentally simulating their behaviors. For example in this set of data, the input, roads and intersections are objects that were considered as part of the (program) structure (S). During designing there were discussions concerning the expected behavior like: drag and drop environment of the input; signal timing of the intersections. An example of behavior derived from structure is the time for cars to travel between intersections.

It may be useful to think of the concepts of independent and dependent variables in a mathematical programming formalism to provide an analogy to the relationships between structure (S) and behaviors (Be and Bs). An independent variable is one that may be varied freely or autonomously by a designer. In a formula that computes area (area = length * breadth), length and breadth are examples of the independent variables. A dependent variable is one that derives from the interaction that occurs between the independent variables, such that its behavior derives from the changes that are effected through the independent variables. Area, therefore, is a dependent behavior that derives from the individual changes that a designer makes to length or breadth. In our coding protocol for the software design example, structure (S) comprises those variables that the designers indicate can be manipulated independently (or conceive that the users of the program may wish to do). Examples are number of lanes, cars or intersection geometry. Behavior comprises those properties that derive from these, for example, the number of cars that can clear through the intersection in a given amount of time.

III. Observations of the Session

In this section we present some basic qualitative and quantitative observations of the protocol session labeled "anonymous" in the NSF Studying Software Designers Workshop protocols. We refer to the two participants as Male 1 and Male 2. Their task was to design a traffic flow simulation program.

A. Qualitative

The software designers spent more than 5 minutes at the beginning to study the brief individually without verbalizing. Male 1 seemed to be more senior and took the leadership role; he started by asking Male 2 to give a summary and how he saw the program structure could be broken down. Male 2 summarized the requirements from the brief. After that, for about three and a half minutes, Male 1 started to externalize on the board. They started with what the required "Data" would be. Then they drew an intersection of a road to discuss the data in relationship to behavior of the simulation program. Different objects/variables were discussed such as cars, signals, intersections, etc.

They started to focus back on the program structure after half an hour and finished the basic layout of the program in

about an hour. Male 1 did most of, if not all, the drawing and writing on the board.

B. Quantitative

The turn taking is very even, Male 1 had 125 utterances and Male 2 had 123 utterances. The word counts of Male 1 and Male 2 are 2777 and 2894 respectively. Figure 2 shows the word count of individuals in near 5 minutes intervals without breaking up the utterances.

The word count shows that the verbal exchange of words were about 516 per five minutes. The word counts varied at the first 30 minutes and dropped to 406 at the 35 minute mark, when they started the discussion on program structure. After that the word count of the verbal exchange started to climb and peaked at the 45 minute mark where the count was 690. After that the word counts declined linearly to 347, at the end of session.

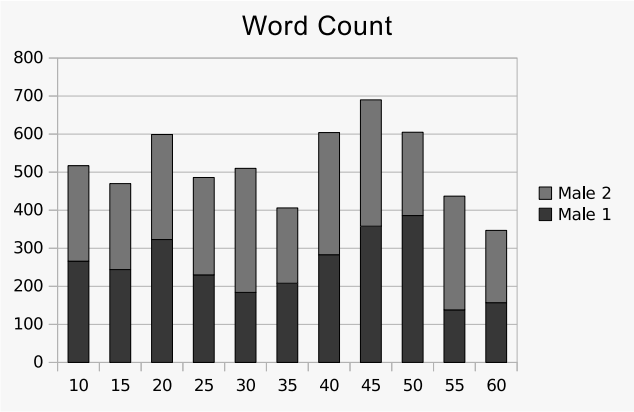

Fig. 2. Word count in five minutes intervals

IV. FBS Segmenting and Coding Scheme

The generic coding scheme applied here consists only of the function (F), expected behavior (Be), behavior derived from structure (Bs), structure (S), documentation (D) and requirement (R). The protocols are segmented strictly according to these six categories. Those utterances that do not fall into these categories will be coded as others (O); these may include utterances related to jokes, social communication, and management.

This framework does not assume any domain of designing nor does it assume the number of participants being studied. It only abstracts and describes the state of affairs of designing in terms of FBS of a segment. With this generic coding scheme, although simple, we aim to capture the essence of designing which will form a foundation for further analysis. Unlike other coding systems, using this coding scheme there is a nexus between segments and codes: one segment per code one code per segment. This forms the basis of segmentation. This produces a higher degree of uniformity and robustness than other methods.

The segmentation/coding involved two independent coders who segmented/coded separately (in different geographical location with no inter-coder bias or influence) and then carried out an arbitration session to produce the final arbitrated segmented/coded protocol. The agreement between the two coders was 82.8% and between the coders and the final arbitrated results was 90.4% respectively. The high percentages of agreement provide additional verification that the FBS ontology based coding methodology provides a logically coherent way to view a protocol by different people. In absence of such a unifying guiding ontological framework, different coders can end up producing widely differing segmentation schemes making any kind of quantitative analysis weak or inaccurate.

V. Results and Analysis

A. Descriptive Statistics

The segmented protocol contains 640 segments, 603 of these are FBS codes, the rest being "O". Figure 3 presents the distribution of the FBS codes. Male 2 refers to the requirements more than Male 1 but overall the "R" coded segments were sparse (17 in total). There were only three function (F) codes (Male 1: 2, Male 2: 1). Male 1 did most of the documentations, where by "documentation" we mean the externalization of thoughts by drawings, symbols and words. The figure shows that "behavior" (300 counts of Be and Bs) is the predominant code. This was 15 times more than the count of the function plus requirement codes (20 counts of F and R); 1.62 times more than the structure (185 counts of S); and 3.13 times more than the documentation codes (96 counts of D).

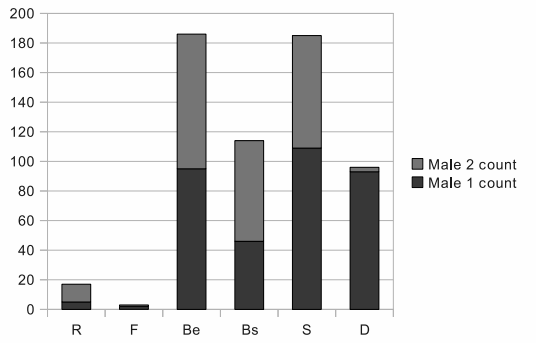

Fig. 3. The distribution of FBS codes

In order to study the change of the designers' behavior, the design session was divided into three equal sized thirds based on the number of segments (1 to 214, 215 to 427, and 428 to 640). Figure 4 shows the change of distribution of the FBS codes among the three divisions.

The "Requirement" and "Function" codes were mainly located at the beginning of the session; only one R code occurred in the middle third and one at the end of the session. There was an increase of expected behavior (Be) and behavior derived from structure (Bs) over time. No trend was observed for structure (S). The externalization (D) decreased.

The decrease of "Requirement" and "Function" mapped well to our understanding and qualitative observation of the

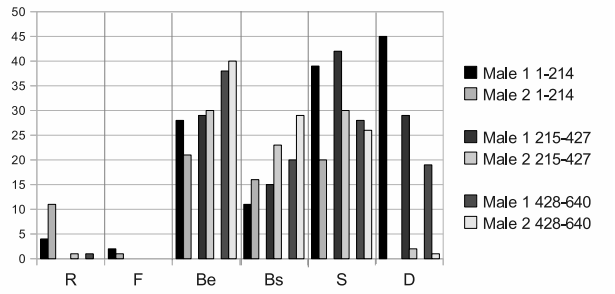

Fig. 4. The FBS code distributions for the beginning, middle and end of the session

session. However, the reduction in "Documentation" was not in accordance with the commonly held view of thinking about the design processes. Reviewing the protocol, the D codes were externalization and did not carry the simple meaning of documentation as producing a final description of a design.

B. Sequential Analysis

Traditionally design protocols were analyzed by static statistical methods that often contain the assumption that each segment is an independent event. Markov chains analyze or describe the probability of one event leading to another. Kan [6] proposed using Markov analysis to examine the sequence of FBS events. Kan and Gero [7] used Markov analysis to compare three design sessions.

Markov chains have been used in a wide variety of domains: to analyze writer's manuscripts and to generate dummy text [8]; for the ranking of web pages by Google [9]; and to capture music compositions and synthesize scores based on the analyzes [10], to name some of its applications.

A probability matrix P is used to describe the transition of the six FBS states (R, F, Be, Bs, S and D). Equation 1 shows the probability transition matrix of the software design session. There are only 3 occurrence of the F state causing four zeros in the row and column of F. This also produces the high probability of F→Be (0.67) because two out of the three consecutive codes that followed F were Be. The occurrence of R is also low, causing two zeros in both the row and column of R.

$$P = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 0.29 & 0.00 & 0.12 & 0.00 & 0.41 & 0.18 \\ F & 0.00 & 0.00 & 0.67 & 0.00 & 0.00 & 0.33 \\ Be & 0.00 & 0.01 & 0.21 & 0.23 & 0.39 & 0.16 \\ Bs & 0.02 & 0.00 & 0.47 & 0.15 & 0.29 & 0.07 \\ S & 0.03 & 0.01 & 0.26 & 0.20 & 0.19 & 0.31 \\ D & 0.01 & 0.00 & 0.42 & 0.15 & 0.43 & 0.01 \end{array}\right) \quad (1)$$

Similar to the count of codes, we divide the design session into three equal sized thirds based on the number of segments. This will allow us to compare the transition behavior during the beginning, middle and end of the session to determine if there are changes in behavior as the session progresses, and P_1, P_2 and P_3 are the probability transition matrices of the segments from 1 to 214, 215 to 427 and 428 to 640 respectively.

$$P_1 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 0.33 & 0.00 & 0.13 & 0.00 & 0.33 & 0.20 \\ F & 0.00 & 0.00 & 0.67 & 0.00 & 0.00 & 0.33 \\ Be & 0.00 & 0.02 & 0.09 & 0.16 & 0.42 & 0.30 \\ Bs & 0.04 & 0.00 & 0.40 & 0.16 & 0.28 & 0.12 \\ S & 0.07 & 0.04 & 0.16 & 0.14 & 0.16 & 0.44 \\ D & 0.03 & 0.00 & 0.40 & 0.18 & 0.43 & 0.00 \end{array}\right) \quad (2)$$

$$P_2 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ F & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ Be & 0.00 & 0.00 & 0.26 & 0.08 & 0.49 & 0.17 \\ Bs & 0.00 & 0.00 & 0.40 & 0.23 & 0.34 & 0.03 \\ S & 0.01 & 0.00 & 0.26 & 0.24 & 0.23 & 0.26 \\ D & 0.00 & 0.00 & 0.37 & 0.17 & 0.43 & 0.03 \end{array}\right) \quad (3)$$

$$P_3 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ F & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ Be & 0.00 & 0.00 & 0.24 & 0.39 & 0.31 & 0.07 \\ Bs & 0.02 & 0.00 & 0.57 & 0.09 & 0.25 & 0.07 \\ S & 0.00 & 0.00 & 0.38 & 0.21 & 0.17 & 0.23 \\ D & 0.00 & 0.00 & 0.53 & 0.05 & 0.42 & 0.00 \end{array}\right) \quad (4)$$

In the middle and end parts of the session, there is no occurrence of F and only one occurrence of R. Examining these three matrices, it is not hard to see the values and the distribution of zeros of P_1 is different from P_2 and P_3. However, it is still hard to comprehend these equations. In the next sub-section, we shall analyze them in terms of the FBS processes.

The mean first passage time M is the average number of steps traversed before reaching a state from other states. The mean passage time can be obtained from the probability matrix. Kemeny and Snell [11] proved that the mean first passage matrix M is given by:

$$M = (I - Z + EZ_{dg})D \quad (5)$$

Where I is an identity matrix, E is a matrix with all entries of 1, D is the diagonal matrix with diagonal elements $d_{ii} = 1/a_i$ where $\alpha = a_1, a_2, ..., a_n$ is the probability vector such that $\alpha P = \alpha$

and Z is the fundamental matrix such that

$$Z = (I - (P - A))^{-1} \quad (6)$$

where each row of A is the same probability vector α and Z_{dg} is the diagonal matrix of Z.

Equation 7 describes the number of state changes expected before reaching a FBS state from other FBS states.

$$M = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 49.1 & 168.4 & 3.9 & 6.7 & 2.5 & 5.0 \\ F & 70.0 & 168.4 & 1.9 & 6.2 & 3.7 & 4.5 \\ Be & 69.3 & 167.3 & 3.2 & 5.0 & 2.7 & 5.3 \\ Bs & 68.3 & 168.2 & 2.5 & 5.4 & 2.9 & 5.8 \\ S & 67.6 & 166.7 & 3.1 & 5.2 & 3.2 & 4.6 \\ D & 68.6 & 168.1 & 2.7 & 5.4 & 2.6 & 6.0 \end{array}\right) \quad (7)$$

Equations 8, 9 and 10 are the mean first passage times matrices of the segments from 1 to 214, 215 to 427 and 428 to 640 respectively.

$$M_1 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 20.1 & 61.6 & 4.4 & 8.3 & 2.8 & 3.6 \\ F & 30.4 & 61.3 & 2.1 & 7.7 & 3.6 & 3.1 \\ Be & 29.6 & 60.0 & 4.1 & 6.7 & 2.6 & 3.2 \\ Bs & 28.8 & 61.3 & 3.1 & 6.8 & 2.9 & 3.8 \\ S & 27.9 & 59.6 & 3.8 & 6.9 & 3.3 & 2.8 \\ D & 29.1 & 61.1 & 3.2 & 6.7 & 2.6 & 4.2 \end{array}\right) \quad (8)$$

$$M_2 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 192.8 & \infty & 4.3 & 6.6 & 1.0 & 6.4 \\ F & 194.2 & \infty & 4.2 & 6.9 & 3.4 & 6.8 \\ Be & 193.9 & \infty & 3.3 & 6.4 & 2.1 & 5.6 \\ Bs & 194.3 & \infty & 2.9 & 5.6 & 2.5 & 6.6 \\ S & 191.8 & \infty & 3.3 & 5.6 & 2.8 & 5.4 \\ D & 194.1 & \infty & 3.0 & 6.0 & 2.3 & 6.5 \end{array}\right) \quad (9)$$

$$M_3 = \left(\begin{array}{c|cccccc} & R & F & Be & Bs & S & D \\ \hline R & 188.8 & \infty & 3.3 & 5.0 & 1.0 & 8.8 \\ F & 187.7 & \infty & 3.3 & 4.8 & 4.4 & 9.8 \\ Be & 187.1 & \infty & 2.6 & 3.3 & 3.3 & 9.1 \\ Bs & 183.8 & \infty & 2.0 & 4.3 & 3.4 & 9.2 \\ S & 187.8 & \infty & 2.3 & 4.0 & 3.7 & 7.8 \\ D & 188.2 & \infty & 2.1 & 4.4 & 2.9 & 9.6 \end{array}\right) \quad (10)$$

There were two columns of ∞ in M_2 and M_3; the values in the R columns were also high in in M_2 and M_3 making these two equations different from M_1. Again, other differences were difficult to observe.

1) Transition Probabilities and FBS Processes: Assuming most of the consecutive events were related and picking the probabilities of those transitions that represent the eight FBS processing in Figure 1, we plot Figure 5.

Formulation only happened at the beginning of the session. The probability of synthesis rose and then dropped. The probability of analysis was low at the beginning. There is a significant increase in the probability of evaluation and type II reformulation at the end of the session.

2) Mean First Passage Time and FBS Processes: Next, with a different assumption – the FBS processes happen not only with consecutive segments. For example when there is a F segment, in order the have a formulation process (F $\rightarrow$ Be), it needs a connected Be segment. With the mean first passage time we can predict when the next Be will appear. With equations 8, 9 and 10 we select those average first passage steps that formed the eight FBS process and charted Figure 6.

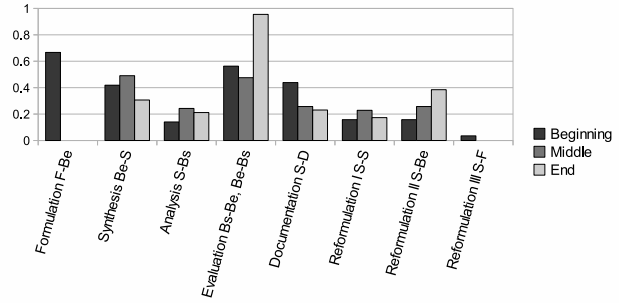

Fig. 5. The eight FBS processes probability distributions for the beginning, middle and end thirds of the session

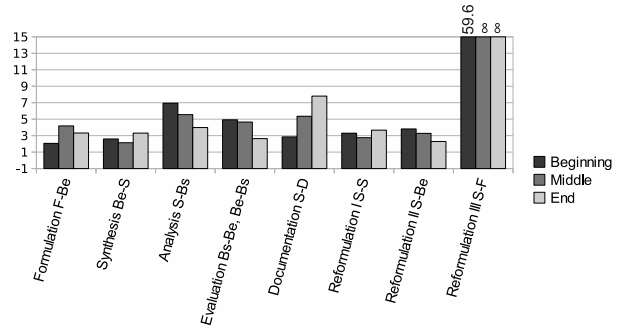

Fig. 6. The distribution of the mean first passage time of the eight FBS processes

Figure 6 shows there is a shift of expecting steps of analysis and documentation when an S segment appeared. At the beginning of the session one can expect an average of 2.8 steps for a documentation process that increased to 7.8 steps at the end of the session. Whereas the expectation of an analysis after an S segment decreased from 5.2 steps to 4.0 steps.

C. Compare with Other Domain of Designing

We compare this software design session with an architectural design session where the task was to generate a conceptual design of a university student union's gallery. The session lasted for 30 minutes. The first 10 minutes was coded for this comparison. Figure 7 compares the code distributions of the first one thirds of the two sessions.

Figure 7 shows that the architectural session had more F, less Be, and more S.

This is an interesting observation: it says that in this architectural design session, the designers focused a lot on function, while in this software design session, they spent very little time on function-based issues. A qualitative analysis confirms this trend. It is well known that in architectural design the same structure configuration can lend itself to a variety of functions. For example, when old buildings in renovated form frequently house a set of activities that the original

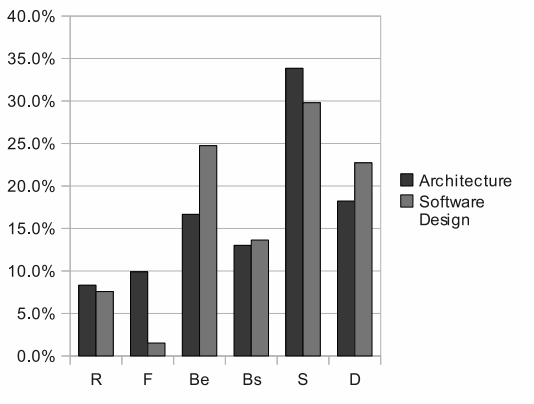

Fig. 7. Comparing the FBS code distribution of the software designing and the architectural designing sessions

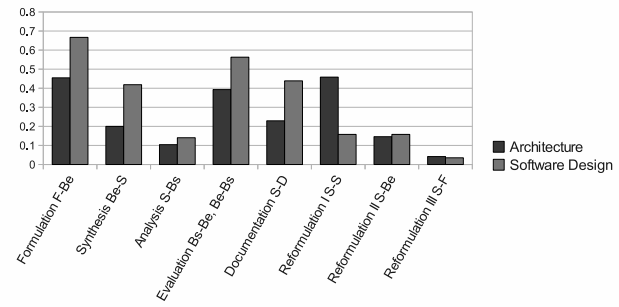

Fig. 8. Comparing the eight FBS processes probability distribution of the software designing and the architectural designing

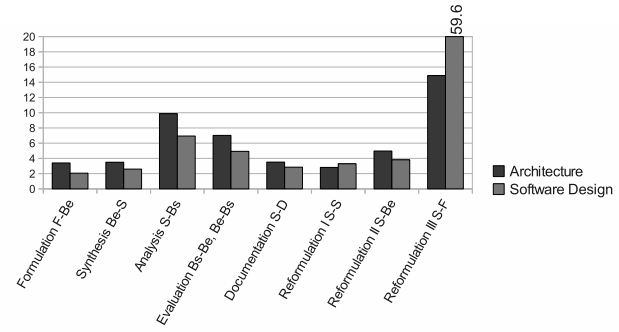

Fig. 9. Comparing the mean first passage time of the eight FBS processes of the software designing and the architectural designing

building was never designed for. So popular is the highly debated relationship between function and form that architects like to debate over whether "function follows form" or "form follows function" with reference to particular contexts in particular design projects. Therefore, in architectural design, the relationship between the set of functions being considered (i.e., the main set of purposes for which the structure is being designed) is an open one, sometimes continuing right through a design session, as architects continue to create or discover new functions, and continue to reformulate and redefine already identified or older ones. In the software design session however, the function space seemed much more defined and fully formulated from the outset. There is frequently one main, explicitly defined function, and the designers' reasoning works a lot more with expected behavior, derived behavior and structure, focused on this defined main function that the program has to fulfill. This discussion shows how the FBS ontological framework provides a powerful common ground from which to analyze designing activity in various domains. It helps to develop rich insights into the similarities as well as the differences in design content, knowledge and processes in different domains through an analysis of "real world" design sessions.

Figure 8 compares the probabilities of the FBS processes of the beginning of the two sessions. The distribution of processes looks similar except for the type I reformulation.

Figure 9 compares the time expectation of the FBS processes of the beginning of the two sessions. The trend looks similar except for the type I and III reformulation.

VI. Conclusion

This paper has applied a generic coding scheme founded on Gero's design-based FBS ontology to the anonymous team of software designers in the NSF Studying Software Designers Workshop protocols. With this generic scheme we have been able to carry out both qualitative and quantitative studies of the design cognition of the designers. As well as compare their behavior to that of architects designing both for different requirements and in a different domain. The FBS coding covers the issues of design interest and the transitions between the codes maps onto design processes. The session was divided into thirds, based on equal number of segments in each third, making a comparisons of the designerly behavior between the beginning middle and end of the session possible. This can be seen in the probability transitions matrices of equations 2, 3 and 4 and in the mean passage times of equations 8, 9 and 10.

An important result of this study is the demonstration that the FBS coding scheme can be used to produce a qualitative and a quantitative understanding of a software design team's designerly behavior.

Another important result of this study is the demonstration that the FBS coding method can be used to compare design activities across domains. In this case study the software designers spent most of the time over behavior-structure relationships of the software being developed, while the architects focused more upon the function-structure-behavior relationships. An overarching similarity is that in both domains, the amount of focus on structure is the highest of all the issues, followed by documentation (externalizing of design reasoning) and expected behavior discussions. The probabilities of formulations and evaluations in the software design is higher than that of the architectural sessions. The distribution of FBS processes looks alike. However, in the architectural design session, the chances of structural reformulation (concerning the form) is much higher. Also, the software design had much longer expectation

time for type III reformulation.

The results also showed that there are differences in designing in terms of the distribution of the FBS design processes between the beginning and the end of the design session.

Considerably more information and knowledge can be generated from the base coded segments. A linkograph [12] can be directly constructed from the segments and this will provide a richer basis for the development of the design processes as the terminals for each link are codes and the transitions from one code to another form the design processes.

ACKNOWLEDGMENT

This research is supported by a grant from the US National Science Foundation grant no. CMMI–0926908.

REFERENCES

[1] N. Cross, H. Christiaans, and K. Dorst, *Introduction: the Delft protocols workshop*. John Wily & Son, 1996, pp. 1–14.

[2] J. McDonnell and P. Lloyd, Eds., *DTRS7 Design Meeting Protocols: workshop proceedings*, 2007.

[3] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[4] J. S. Gero, "Design prototypes: a knowledge representation schema for design," *AI Magazine*, vol. 11, no. 4, pp. 26–36, 1990.

[5] J. Gero and U. Kannengiesser, "The situated function-behaviour-structure framework," *Design Studies*, vol. 25, no. 2, pp. 373–391, 2004.

[6] W. T. Kan, "Quantitative methods for studying design protocols," Ph.D. dissertation, The University of Sydney, Faculty of Architecture, Design & Planning, Key Centre of Design Computing and Cognition, 2008.

[7] J. W. T. Kan and J. S. Gero, "A generic tool to study human design activity," in *Human Behavior in Design*, M. Norell Bergendahl, M. Grimheden, L. Leifer, P. Skogstad, and U. Lindemann, Eds., 2009, pp. 123–134.

[8] H. Kenner and J. O'Rourke, "A travesty generator for micros: Nonsense imitation can be disconcertingly recognizable," *BYTE*, no. 12, pp. 449–469, 1984.

[9] A. N. Langville and C. D. Meyer, "Updating markov chains with an eye on google's pagerank," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 4, pp. 968 – 987, 2006.

[10] M. Farbood and B. Schoner, "Analysis and synthesis of palestrina-style counterpoint using markov chains," in *Proceedings of International Computer Music Conference*, Havana, Cuba, 2001.

[11] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, ser. The University Series in Undergraduate Mathematics. D. Van Nostrand Company, Inc. Princeton, New Jersey, 1960.

[12] G. Goldschmidt, *Criteria for design evaluation: a process-oriented paradigm*, ser. Principles of Computer-Aided Design. John Wiley & Son, Inc., 1992, pp. 67–79.