Kong, S.C., Ogata, H., Arnseth, H.C., Chan, C.K.K., Hirashima, T., Klett, F., Lee, J.H.M., Liu, C.C., Looi, C.K., Milrad, M., Mitrovic, A., Nakabayashi, K., Wong, S.L., Yang, S.J.H. (eds.) (2009). Proceedings of the 17th International Conference on Computers in Education [CDROM]. Hong Kong: Asia-Pacific Society for Computers in Education.

Constraint-based Design Critic for Flat-pack Furniture Design

Yeonjoo OH[a], Mark D GROSS[a], Suguru ISHIZAKI[b], Ellen Yi-Luen DO[c]
[a]*CoDeLab, Carnegie Mellon University, USA*
[b]*Department of English, Carnegie Mellon University, USA*
[c]*Colleges of Architecture and Computing, Georgia Tech, USA*
yeonjoo@cmu.edu

Abstract: This paper reports on the Flat-pack Furniture Design Critic (FFDC). By analyzing the literature of architecture education, we have identified critiquing methods: delivery types (interpretation, introduction, example, demonstration, and evaluation) and communication modalities (written comments, graphical annotations, and images). Our FFDC uses these methods to deliver feedback. This paper also presents how our FFDC system selects particular methods by considering a certain condition such as user's knowledge level and the previously used methods.

Keywords: design critiquing, constraint-based tutors, delivery types, modalities

Introduction

The studio occupies a pedagogically important position in design education. It is the main academic course in any architecture or industrial design program. Students in a studio are subjected to a series of critiquing sessions, where instructors offer critiques on their work. Essential among these sessions is the "desk crit"—a one-on-one critiquing session [1]. For desk crits, an instructor visits individual students' places to critique their work, while other students wait their turn. This long lineup of students limits the time the instructor can spend on each student, often resulting in a curtailed or superficial discussion [2]. To address this problem, we envision that a computer program that could offer effective critique could help individual students learn designing just as intelligent tutoring systems (ITS) help students learn other subjects such as algebra [3].

Our main goals in this work are first, by reviewing the literature of architecture education, to identify what critiquing methods studio teachers use; and second, to develop a computer-based design critic that incorporates these methods, choosing particular methods based on conditions such as a student's knowledge level and the methods that the system has previously used with the student. This paper presents our Flat-pack Furniture Design Critic (FFDC) system as a step toward creating computer-based critics that support design learning in studio settings. Our FFDC program supports multiple critiquing methods.

1. Related Work

1.1 Constraint-based Tutors (CBT)

Our FFDC program adopts the typical system architecture of constraint-based tutors, which is suited for design domains. An often-cited characteristic of design is that it lacks well-structured domain models and that a design problem seldom has a single or best solution. In this domain, no systematic way exists to determine when a proposed solution is acceptable. We chose constraint-based tutors, because this approach does not require a complete domain model. Constraint-based tutors model domain knowledge using a set of constraints that specify what characteristics a solution should or should not have. These constraints can provide only a partial description of a solution. The effect of a missing constraint is highly restricted, resulting only in failing to detect a particular error. A proposed solution can still be analyzed with other constraints. Thus, we can develop a domain model incrementally.

These constraint-based tutors derive from Ohlsson's theory of *learning from performance errors* [4]. Ohlsson argues that learning occurs when students catch mistakes by themselves or when others catch mistakes for them. The fundamental assumption is that certain problem states reveal diagnostic information. This assumption starts from the fact that one cannot develop acceptable solutions that violate domain principles. Antonija Mitrovic and her Intelligent Computer Tutoring Group (ICTG) has explored various topics in constraint-based tutoring, for example, supporting a variety of tasks, enhancement of student models, new strategies to deliver feedback, and development of authoring systems [5].

Each constraint represents a piece of domain knowledge; it consists of a relevance condition and a satisfaction condition. The relevance condition indicates when the constraint should apply—and the satisfaction condition represents states where a certain piece of knowledge has been correctly applied. Therefore, a solution must satisfy the satisfaction condition, when the constraint is deemed relevant to the user's solution. For example, a constraint for designing a chair for writing, whose seat's height range must be 380*mm* – 510*mm*, could be written as: If <designing a chair for writing>, then <the seat height above floor must be more than 380*mm* and less than 510*mm*>. A violated constraint indicates an opportunity to improve the proposed design so the tutor offers feedback regarding the violated constraint.

Some constraint-based tutors, for example, Kermit [6] record information about a student to deliver individually tailored instruction. This *student model* consists of the history of all constraints that the tutor has applied to the student's design, including both satisfied and violated constraints. The violated constraints indicate domain knowledge the student has not yet mastered. Based on this diagnosis, the constraint-based tutor provides feedback to help the student improve the solution.

1.2 Critiquing Methods used in Intelligent Tutoring Systems and Critiquing Systems

Conventional intelligent tutoring systems and critiquing systems do not provide feedback using the rich range of methods that design instructors employ in studio teaching. (These are outlined in the following section.) Most computer-based systems use negative evaluation to provide feedback, merely pointing out problems or errors. Although researchers in intelligent tutoring and critiquing have explored various ways to interact with users such as argumentation [7], examples [8], dialogue [9], question-asking [10], or self-explanation [11], individual systems still do not cover all critiquing methods that studio instructors are using. Several systems also offer feedback using multiple modalities. For example, Reading Tutor [12] combines speech and graphics (highlighting); AutoTutor [9] combines speech with 3D simulation and facial expression; Design Evaluator [13] combines text with graphical annotation of a 3D model; and KID [8] combines text with images. Although taken together these systems recognize diverse methods to interact with users, *we are unaware of any single system that makes decisions, based on a student model, about when to use which method to deliver particular critiquing methods*. That is the focus of the system we present here.

2. Critiquing Methods – Delivery Types and Communication Modalities

Studio instructors in architectural design use a variety of critiquing methods to convey their knowledge and professional skills. We divide these methods into two categories: delivery types and communication modalities.

2.1. Delivery Types

Uluoglu [14] and Bailey [15] both identify diverse 'delivery types' by analyzing critiquing sessions in architecture studios. These include (1) ***interpretation*** of the students' design solutions, (2) ***introduction*** of new ideas or approaches, (3) description of existing ***examples*** or precedents, (4) ***demonstration*** of potential solutions or other design actions, and (5) ***evaluation*** (positive or negative) of the students' solutions. The choice of delivery types is important, because it may influence the students' subsequent actions and hence their learning. For example, when a teacher offers existing examples, students may look at how the given examples develop ideas and attempt to apply the ideas to their solutions. On the other hand, when the teacher points out errors, the students may fix these errors. Table 1 shows examples of delivery types in an architectural design critique [16].

Table 1 Feedback Instances of Five Delivery Types

Delivery Types	Feedback Instances
Interpretation	"Your building is just getting light into this level (pointing to the bottom window on the physical model)"
Introduction	"Have you thought about the sun's path over a day and a year?"
Example	"Le Corbusier's building has a similar concept. Look at the windows of his chapel at Ronchamp."
Demonstration	"You need to make a form here. You need to do something here (drawing a line that represents a wall)"
Evaluation	(positive) "You take the rough form into something more precise.… which is good" (negative) "No good, horrible—it just ruins the whole idea."

2.2. Communication Modalities

We define 'communication modalities' as channels such as speech, text, and drawing. The primary modality in all face-to-face critiquing sessions is speech—teachers always talk. Studio teachers also make brief notes as they draw, or annotate their students' sketches. Although these notes are terse, they help students remember the spoken feedback. Design teachers often use drawings, ranging from abstract diagrams to representational forms. Schön [16] and Anthony [2] both note that critiques presented in multiple modalities work together and help students understand the intentions of their instructors (see Figure 1).

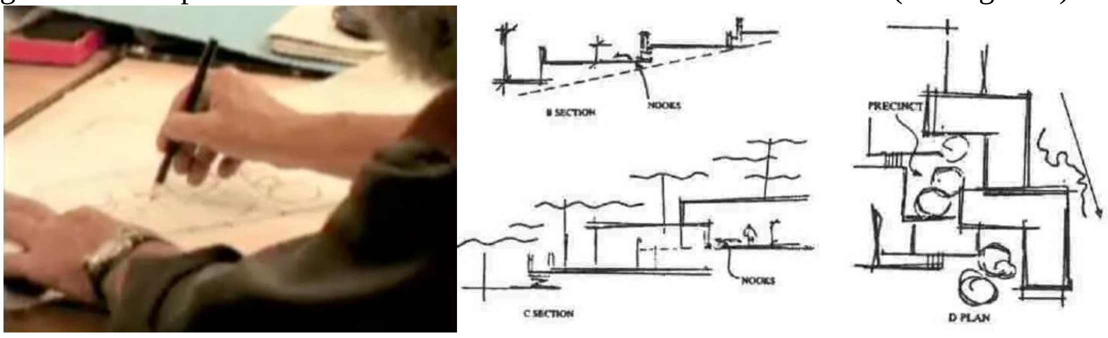


Figure 1 Communication Modalities: (a) A studio instructor, Jan Wampler, sketches on a student's drawing, while offering feedback verbally (Source: MIT Open Courseware [17]); (b) a studio instructor makes sketches and brief notes (Source: Schön's The Design Studio [16])

2.3. Multiple Critiquing Methods

Studio instructors offer feedback using multiple critiquing methods [18] to deliver images, ideas, examples, and actions acquired from their experiences. The instructors have accumulated their own collections of images, ideas, examples, and actions. Schön [18] calls this collection a '*repertoire*'. When instructors look at a student's solution, they scan their repertoires and for similar situations, for example, buildings they have known, or problems they have previously encountered. The instructors not only point out errors; they also describe examples or demonstrate how to solve the problems. Feedback presented using multiple methods helps design students understand their problems better, eliminate errors from their proposed solutions, and construct their own repertoires [14, 18].

3. Flat-pack Furniture Design Critic (FFDC)

Inspired by the richness of critiquing in architectural design studio, we have built a constraint based design critic program that offers students feedback using five delivery types (interpretation, introduction, example, demonstration, and evaluation) and three communication modalities (written comments, graphical annotations, and images). Our Flat-pack Furniture Design Critic (FFDC) selects delivery type and modality to present a critique using a model of the student's task and the criticism that the student has previously received.

Our program utilizes the identified critiquing methods used in architecture studios. However, we chose as our test-bed for system development the simpler design domain of flat-pack furniture instead of architecture for several reasons. First, although the problem space of flat pack furniture design is relatively small, it is still ill-defined and open-ended. Second, furniture design is often used as an early exercise for first-year architecture students. Finally, furniture designers familiarize themselves with design problem-solving by drawing and modeling in the same ways as architecture students do.

3.1. System Architecture

Our FFDC is written in MCL (Macintosh Common Lisp) using OpenGL to provide 3D models and the Lisa (Lisp-based Intelligent Software Agent) production rule system to reason about a proposed furniture design using the stored constraints. FFDC comprises a number of components: it has Construction Interface, Parser, Pattern Matcher, Design Constraints, Critiquing Rules, User Model, Pedagogical Module, and Critiquer. Figure 2 shows these components, their relationships, and the information flow among them.

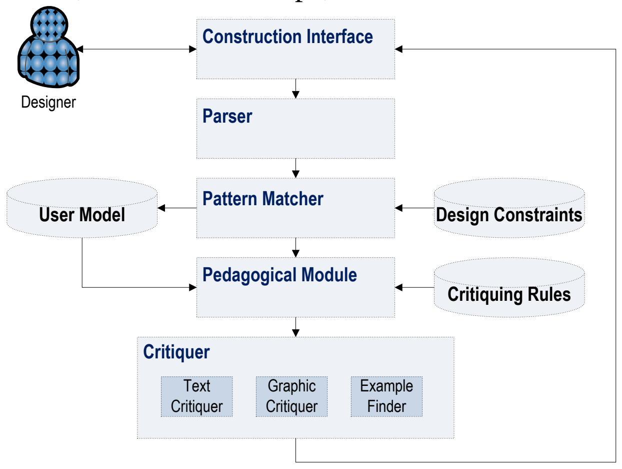


Figure 2 Main FFDC Components in the Iterative Construction-Critiquing-Repair Cycle.

A designer starts designing by sketching an axonometric diagram in the *Construction Interface* using a stylus and a digitizing tablet. The program records all sketched glyphs, identifies the Cartesian coordinate system, and generates a 3D model (shown in Figure 4-(a) and (b)).

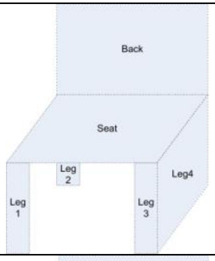
The *Parser* parses the sketched diagram and the 3D model, producing two kinds of data: parts and their properties (e.g., x-length, plane, 3D coordinate data, joints, etc.) and configuration of parts (e.g., parallel, between, top-of, jointing, distance, etc.). The *Parser* creates a text file to store a symbolic representation of the designed furniture.

The program stores a set of *Design Constraints* that represent principles that designers need to know. FFDC uses two types of constraints: 27 structural constraints that specify forbidden/allowed structures of furniture parts and 36 functional constraints that specify allowed functions of certain parts or a whole piece of furniture.

The *Pattern Matcher* compares the symbolic representation of the design against the *Design Constraints* in order to detect critiquing opportunities. For example, a chair design in Figure 4-(a) violates the stored constraint that 'a chair must have armrests'. The following pseudo-code and diagrams show the constraint that the design has violated.

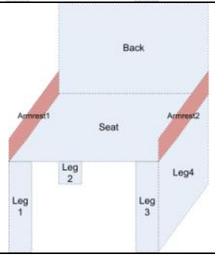
```
If the designed furniture is a chair

((Seat is placed on top of Leg1)
 (Seat is placed on top of Leg2)
 (Seat is placed on top of Leg3)
 (Seat is placed on top of Leg4)
 (Back is placed on top of Seat))
```



```
then the chair must have armrests

((Armrest1 is placed top of Seat)
 (Armrest2 is placed top of Seat)
 (Armrest1 is placed on left side of Back)
 (Armrest1 is jointed with Back)
 (Armrest2 is placed on right side of Back)
 (Armrest2 is jointed with Back))
```



FFDC stores two types of *User Model*: a short-term and a long-term user model. The short-term user model stores the reasoning outputs of the *Pattern Matcher*, namely the violated and satisfied constraints for the current critiquing session. Each violated/satisfied constraint stores (1) the unique constraint number to indicate which constraint is violated or satisfied; (2) whether this constraint is violated (V) or satisfied (S); (3) how many times this constraint has been violated or satisfied; (4) what furniture parts violate/satisfy this constraint; (5) the critiquing delivery types that have already been used to offer feedback on this constraint; and (6) the used critiquing communication modalities. The long-term user model stores the history of all violated and satisfied constraints over multiple critiquing sessions. Using this history of all constraints, the program makes inferences about (1) how much a designer knows about this flat-pack design field; (2) the specific strengths and weaknesses of the designer; and (3) which critiquing method works well for a certain designer. For example, the program identifies a designer who tends to violate important constraint as a novice. It also observes which types of constraints a designer tends to violate. When the designer mainly violates structural constraints, the program concludes that the designer is weak in structural knowledge.

3.2. Selecting Particular Delivery Types and Communication Modalities

This section describes how the FFDC selects particular set of delivery types and modalities. It explains (1) our constraint design, and (2) the *Pedagogical Module* and the *Critiquing Rules*.

3.2.1. Our Constraint Design

Each constraint data structure has two slots relevant to offering feedback in multiple methods: *critique-delivery-types*, and *critique-modalities*. The *critique-delivery-types* slot stores pre-defined written comments for the constraint in five different delivery types. For example, a bookcase design (Figure 4-(b)) violates a constraint that checks whether a back part is large enough to support lateral loads. The *critique-delivery-types* slot stores written comments in five different delivery types:

```
((Interpretation – "Your bookcase is composed of two sides, a shelf, a top and a back")
```

(Introduction – "Do you think that your back part is big enough to support lateral loads?")
(Example – "Please see how other furniture support lateral loads from the shown examples")
(Demonstration – "You need to make the back part bigger as shown")
(Evaluation – "Your furniture is structurally unstable to support lateral loads"))

The *critique-modalities* slot stores a list of calls to routines that deliver feedback in different communication modalities. The FFDC delivers feedback using the selected communication modalities by executing these routines: graphic annotations, e.g., painting parts that violate a constraint in red; displaying graphic icons such as arrows to indicate load placed on a furniture part; and retrieving and presenting images of relevant examples.

3.2.2. Pedagogical Module and Critiquing Rules

The *Pedagogical Module* takes as input (1) the data of a violated constraint from the short-term user model, and (2) the data about a specific designer from the long-term user model. It then chooses particular critiquing methods by applying the *Critiquing Rules* (see Figure 3). The FFDC has a set of *Critiquing Rules* that specify which delivery types and communication modalities to use under what conditions. In other words, the *Pedagogical Module* considers the violated constraint and the designer's history with the system in order to select a particular critiquing method (Figure 3).

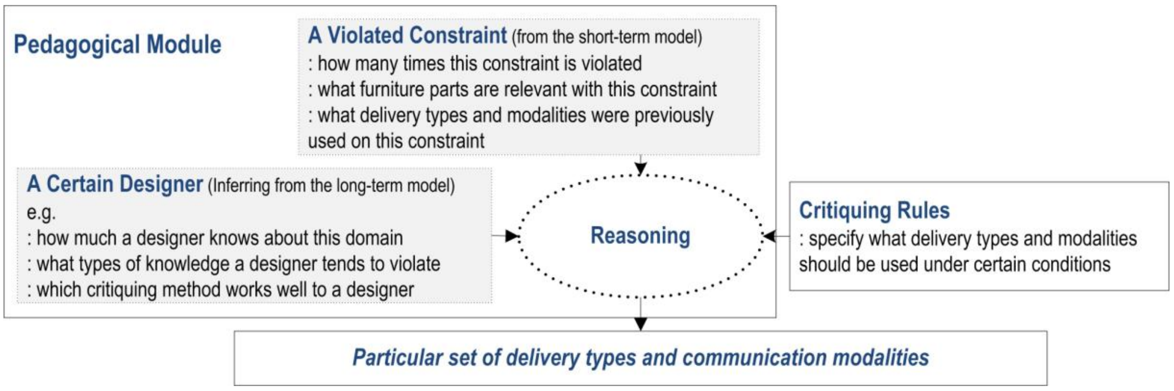

Pedagogical Module
A Violated Constraint (from the short-term model)
: how many times this constraint is violated
: what furniture parts are relevant with this constraint
: what delivery types and modalities were previously used on this constraint
A Certain Designer (Inferring from the long-term model)
e.g.
: how much a designer knows about this domain
: what types of knowledge a designer tends to violate
: which critiquing method works well to a designer
Reasoning
Critiquing Rules
: specify what delivery types and modalities should be used under certain conditions
Particular set of delivery types and communication modalities

Figure 3 Pedagogical Module and Critiquing Rules

Our FFDC system selects critiquing methods differently according to certain conditions. When the program knows nothing about the designer, or if the long-term user model stores no history of the violated constraints, the *Pedagogical Module* chooses delivery types and modalities following two sequences: (1) interpretation – introduction – example – demonstration – evaluation, and (2) written comments – graphic annotations – images – multiple modalities. These two sequences are the initial setup for selection of critiquing methods. Once the program learns more about the designer, it selects delivery types and modalities following the *Critiquing Rules*. For example, when a designer is identified as a novice, the *Pedagogical Module* will select the 'demonstration' delivery type rather than 'example' because novices often have difficulty utilizing examples in their designs. Or if a designer tends to violate structural constraints, the *Pedagogical Module* will select graphical annotation with written comments, because feedback in multiple modalities (text + drawing) works better for a student who lacks prior knowledge about the subject matter [19].

3.3. Presenting Feedback using the Selected Delivery Types and Modalities

Once the *Pedagogical Module* selects a critiquing method, the *Critiquer* activates one or more of its components to present the critique to the designer. The *Critiquer* has three components: (1) a Text Critiquer, which presents the written comments associated with a violated constraint, (2) an Example Finder, which selects relevant examples from a library, and (3) a Graphic Critiquer, which highlights relevant furniture parts and draws graphical annotations on a designer's diagram. If the *Pedagogical Module* selects the critiquing methods 'introduction' and 'graphical annotation', then the *Critiquer* activates two components (Figure 4-(a)): the Text Critiquer and the Graphic Critiquer. The Text Critiquer presents the stored 'introduction' message from the violated constraint. The Graphic Critiquer executes function calls stored in the *critiques-modalities* slot (using the stored relevant furniture parts as parameters) to annotate the designer's diagram.

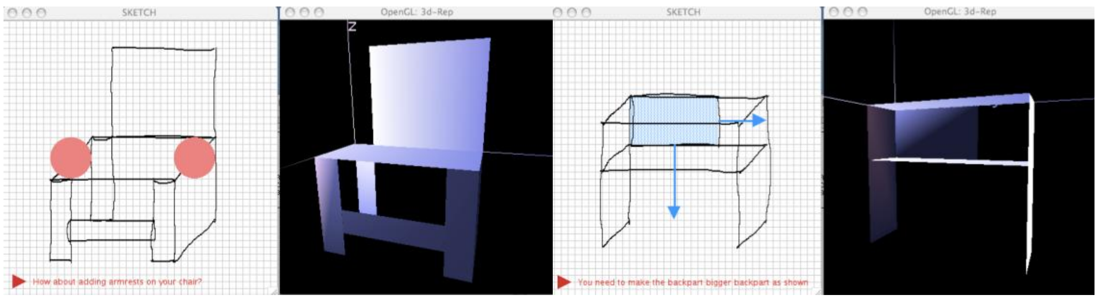

SKETCH
OpenGL: 3d-Rep
Z
How about adding armrests on your chair?
SKETCH
OpenGL: 3d-Rep
You need to make the backpart bigger backpart as shown

Figure 4. (a) Chair: FFDC ***introduces*** a new idea of adding armrests to a chair for a user's comfort by making graphic annotations in the Construction Interface (two circles) to indicate possible positions of armrests and displaying written comment "*How about adding armrests to your chair?"* (introduction + graphical annotation); (b) **Bookcase**: FFDC ***demonstrates*** how to resolve the detected situation by making graphical annotations in the Construction Interface (a rectangle with two arrows) and displaying a written comment *"You need to make the back part bigger as shown"*(demonstration + graphical annotation).

When the *Pedagogical Module* selects 'example' as delivery type and 'images' as modality, the *Critiquer* activates the Text Critiquer and the Example Finder. The Example Finder looks through stored designs that other designers have made and retrieves the relevant ones. The program stores a furniture design as three kinds of data in a text file: (1) the parsed data that the *Parser* has generated, (2) the geometrical data of the drawn diagram and (3) a list of the violated constraints of the design (when constraints are violated). The Example Finder compares previously stored designs with the current design to retrieve relevant cases. For example, suppose that a certain design that the Example Finder finds is a chair. The Example Finder scans through previously saved files. If it finds that the parsed data in a file satisfy the constraint that checks whether a design is a chair, it decides the retrieved design is relevant. The *Critiquer* then presents all the retrieved chairs (Figure 5).

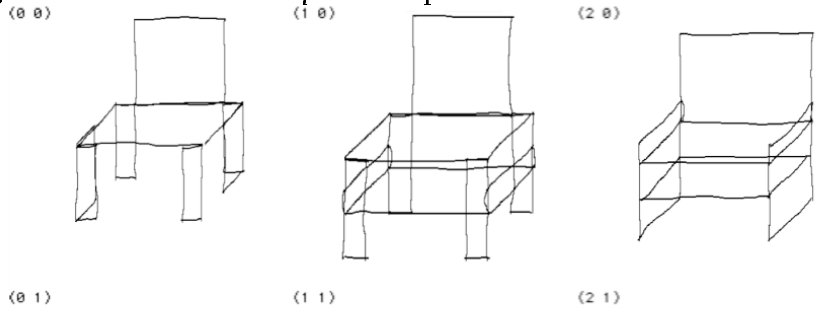


Figure 5 FFDC Presents Several Examples of Other Chairs

4. Conclusion

The Flat-pack Furniture Design Critic applies the constraint based tutoring approach to the domain of design. We also have analyzed design critiquing by reviewing the literature of the pedagogy of architecture studios. FFDC supports delivery types and communication modalities that are used in architectural education settings. It also selects particular methods to deliver feedback by considering a user's knowledge and the critiquing methods that the program has previously used for this user.

Our FFDC system is intended to close the gap between human critics and computer-based critiquing systems and intelligent tutors. It adds the richness of design critiquing to the conventional feedback of those systems in the form of diverse delivery types and communication modalities. We believe that feedback presented in these multiple methods can help designers develop their solutions better and learn designing better.

We chose flat-pack furniture designing as an example domain, but our system could be extended for other domains such as architecture, product design, or engineering. Our system mechanism that selects particular critiquing methods is applicable to other domains: The system determines critiquing methods by considering domain independent information such as a user's knowledge level and previously used delivery types and modalities. A system designer could implement a system for another domain by developing a *Parser* that analyzes designs and *Design Constraints* that represent domain knowledge.

FFDC could also serve as a tool for experimenting with different delivery types and communication modalities for learning design. These experiments could help us refine and enhance the selection mechanism of critiquing methods to support learning design.

References

[1] Goldschmidt, G. (2002). One-on-One: A Pedagogic Base for Design Instruction in the Studio Common Ground Design Research Society International Conference Brunel University 430 - 437

[2] Anthony, K. H. (1991). Design Juries on Trial: the Renaissance of the Design Studio. New York: Van Nostrand Reinhold.

[3] Koedinger, K. R., Anderson, J. R., Hadley, W. H., & Mark, M. A. (1997). Intelligent Tutoring Goes to the Big City. International Journal of Artificial Intelligence in Education, 8, 30 - 43.

[4] Ohlsson, S. (1996). Learning from Performance Errors. Psychological Review, 3(2), 241 - 262.

[5] The Intelligent Computer Tutoring Group (ICTG). (2009). Retrieved Aug, 20, 2009, from http://ictg.canterbury.ac.nz/

[6] Suraweera, P., & Mitrovic, A. (2002). Kermit: A Constraint-Based Tutor for Database Modeling. Intelligent Tutoring Systems, 2363, 377 - 387.

[7] Fischer, G., McCall, R., & Morch, A. I. (1989). Design Environments for Constructive and Argumentative Design. Human Factors in Computing Systems (CHI '89), Austin, Texas. 269 - 275.

[8] Nakakoji, K., Yamamoto, Y., Suzuki, T., Takada, S., & Gross, M. D. (1998). From Critiquing to Representational Talkback: computer support for revealing features in design. Knowledge-Based Systems, 11(7-8), 457 - 468.

[9] Graesser, A. C., Chipman, P., Haynes, B. C., & Olney, A. (2005). AutoTutor: An Intelligent Tutoring System with Mixed-initiative Dialogue. IEEE Transactions in Education, 48, 612 - 618.

[10] Milik, N., Marshall, M., & Mtrovic, A. (2006). Responding to Free-form Student Questions in ERM-Tutor. Lecture Notes in Computer Science, 4053, 707 - 709.

[11] Mitrovic, A. (2002). NORMIT: A Web-Enabled Tutor for Database Normalization. International Conference on Computers in Education (ICCE). 1276 - 1280.

[12] Mostow, J., Aist, G., Burkhead, P., Corbett, A., Cuneo, A., Eitelman, S., et al. (2003). Evaluation of an Automated Reading Tutor that Listens: Comparison to Human Tutoring and Classroom Instruction. Journal of Educational Computing Research, 29(1), 61 - 117.

[13] Oh, Y., Do, E. Y.-L., & Gross, M. D. (2004). Intelligent Critiquing of Design Sketches. AAAI (American Association for Artificial Intelligence) Fall Symposium - Making Pen-based Interaction Intelligent and Natural, Washington DC. 127 - 133.

[14] Uluoglu, B. (2000). Design Knowledge Communicated in Studio Critiques Design Studies, 21(1), 33 - 58

[15] Bailey, R. O. N. (2004). The Digital Design Coach: Enhancing Design Conversations in Architecture Education. PhD Dissertation Victoria University of Wellington

[16] Schön, D. A. (1985). The Design Studio. London: RIBA.

[17] Wampler, J. (2002). Architecture Studio: Building in Landscapes. Retrieved Jan. 2009, from MIT Open Courseware: http://ocw.mit.edu/OcwWeb/Architecture/4-125Architecture-Studio--Building-in-LandscapesFall2002/CourseHome/index.htm

[18] Schön, D. A. (1983). The Reflective Practitioner: How Professionals Think in Action Basic Books Inc. .

[19] Mayer, R. E. (2001). Multimedia Learning: Cambridge University Press.