

DESIGN BY SEARCHING:
A SYSTEM FOR CREATING AND EVALUATING COMPLEX
ARCHITECTURAL ASSEMBLIES

by

Matthew Giles Phillips
Bachelor of Architecture
Virginia Polytechnic Institute and State University, 1999

SUBMITTED TO THE DEPARTMENT OF ARCHITECTURE IN PARTIAL
FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN ARCHITECTURE STUDIES
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MAY 2007

Copyright (c) 2007 Matthew Giles Phillips. All Rights Reserved.

The author hereby grants to MIT permission to reproduce
and distribute publicly paper and electronic
copies of this thesis document in whole or in part.

Signature of Author: _____

Department of Architecture

May 23, 2007

Certified by: _____

Kent Larson

Principal Research Scientist in Architecture

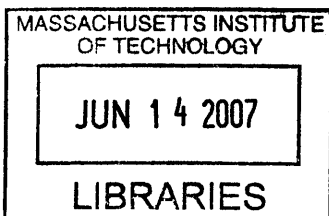
Thesis Supervisor

Accepted by: _____

Julian Beinart

Professor of Architecture

Chairman, Committee for Graduate Students



ROTC

Matthew Giles Phillips

Design By Searching

A System for Creating and Evaluating Complex
Architectural Assemblies

Readers:

Dr. Terry W. Knight

Dr. George Stiny

Dr. William J. Mitchell

Acknowledgements

To Kent Larson for providing both the opportunity to pursue this work and for your tireless guidance along the way.

To Terry Knight for your tremendous assistance and support in shaping this research, you have been a guide for me during my time at MIT.

To George Stiny for your thoughtful and practical suggestions.

To Bill Mitchell for your valuable perspective on the issues at the heart of this work.

To my participants for thier time and energy, and for contributing to so much of what this research has to offer.

To Carla Farina for your assistance in preparing numerous materials.

To my family, for believing in me.

To the 32 students lost in Blacksburg, Virginia, April 16th, 2007, may you rest in peace.

I would also like to thank:

Dr. Stephen Prince

Neeraj Bhatia

Lu Ai

Jennifer Beaudin

Kenneth C. Cheung

Christoforos Romanos

Michael Fadel

Contents

Design by Searching

A System for Creating and Evaluating Complex
Architectural Assemblies

1 Abstract	9
2 Introduction	10
3 Design by Searching	16
4 The Result Set: Component Based Representations	37
5 The Query: Conceptual, Constructive Interfaces	45
6 User Evaluations and Findings	74
7 Search System Prototype	96
8 Where Do We Go From Here?	125

APPENDICES

Appendix 1: User Exercises 133

Appendix 2: System Details 153

References 168

Abstract

Design by Searching

A System for Creating and Evaluating Complex Architectural Assemblies

This work investigates a prototypical Web-based search system designed to enable architects and/or developers to engage and educate residential consumers in a new way: as co-designers. The key motivation is to develop software tools that support a feasible industrial process while providing home consumers with a way to conceive of and design spaces, as an alternative to the standardized commodity solutions that are currently available. The basic mode of operation for this work is to combine the structure of the modern computational search with emerging building modeling technologies as a foundation for Web-based participative design tools.

Object-oriented component representations have been utilized to build a solution space that can be searched directly, without indexing. Additionally, conceptual query interfaces have been designed and evaluated through interviews with volunteer users. The component-based solutions and conceptual queries were then incorporated into a prototype of an architectural search tool which was analyzed to measure its effectiveness.

Thesis Supervisor:

Kent Larson

Principal Research Scientist in Architecture

Readers:

Terry W. Knight

George Stiny

William J. Mitchell

1 *Introduction*

Architects don't have a meaningful role in the design of most homes. They design the usual avant-garde houses that decorate magazines, and the custom one-off designs that get built for rich people, but these and the other projects that involve architects comprise only a small number of the homes that get built in the US. The rest of our houses do not involve architects in developing a solution for individual homeowners, and certainly do not involve the potential homeowners in the design – they are standardized in design to target the center of the home market. This is, of course, a result of the economics of the home market and the developer's production methods: by building standardized, non-variant houses or living units, developers reduce complexity and thus reduce cost. But these homes do not have any direct relation to the potentially changing needs of individuals; the idea is that the standard templates are good enough. The process endures because the people have no vocal chords for expression and thus no voice for complaint – houses are expensive and the process of ownership is complicated and intimidating. Plus, non-standard homes may be difficult to sell, being anomalies in the housing market: wanted by a few people perhaps, if only those people could find them. So the market is built upon our silent acceptance of mass-standardization, because through our limited choice and limited awareness of the market, we the people let it be so.

In the market-driven world of housing development, is there any space for the dreams of humanity in the building process? Can people engage with architects and design on a large scale? Perhaps there is a way. In looking at current trends in the industry, advances in technology may become a facilitator not only for new design processes but for consumer-participative design processes. In computers, the recent emergence of component-based representations for architectural structure means that software can now know more about the spaces upon which it operates. For example: Autodesk, the market leader for CAD tools, has shifted from their traditional AutoCAD software platform to Revit, which utilizes Building Information Modeling (BIM), a component-based representation.

Their goal in making this shift was to better facilitate information exchange between architects, builders, and engineers. And the new, more detailed computational representations like BIM are already showing the potential to transform the home production process, essentially reducing economic barriers that worked architects out of the loop in the first place, by making it easier to share and communicate and by reducing field labor costs. Reconfigurable, pre-fabricated assemblies, described as a hierarchy of components, can be made and assembled affordably, and described computationally, which may facilitate mass-custom design on a higher level.

This work explores the potential of emergent technological innovations to facilitate a democratization of the design process, where interested home-buyers can interact with high-level tools that are built on top of more descriptive representations like BIM – representations which are in turn connected to fabrication processes, thus realizing a new model for mass-customization. One of the key challenges is to find high level representations that allow people to understand and design spaces, according to their own needs and values. Apart from a plethora of web-based, traditionally searchable listings, there are two broad categories of home design tools that are currently available to people. These are simplified narrative “configurator” type tools, and tools that are based upon a 2D or 3D modeling environment. The former is so constrained that truly rich preference specification simply isn’t possible, and the latter is very unfamiliar to most users.

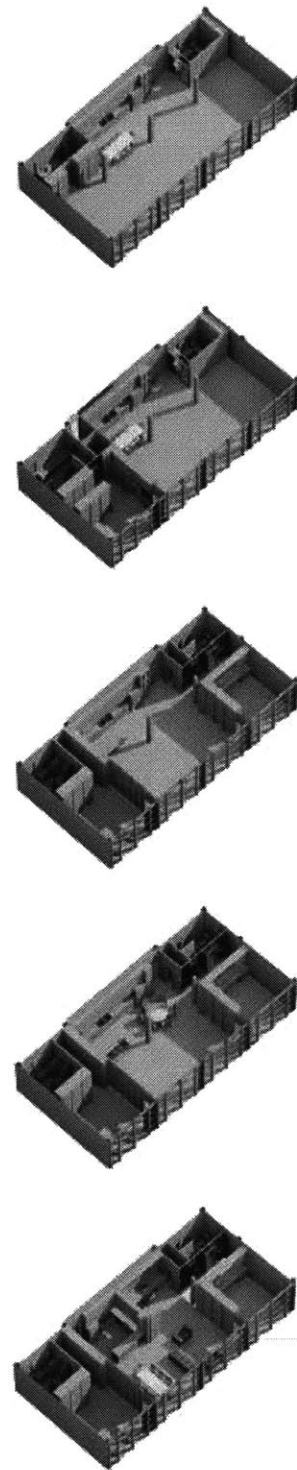
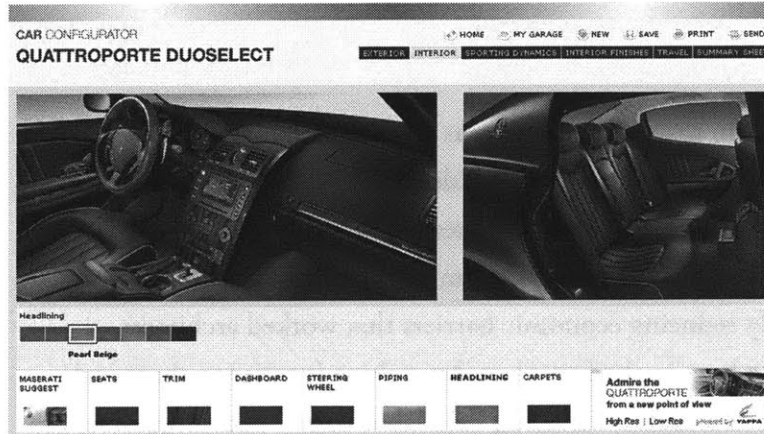


Figure 1.1 Different Apartment Layouts
in a component-based representation
K Larson, M G Phillips, C Farina
rendering by C Farina

Figure 1.2 Configurator System
An example from the automotive industry,
the Maserati car configurator
<http://www.maserati.com>



Configurator tools (Figure 1.2) are basically narrative scripts that combine different types of representations like text and images to describe the design product to people. As will be described, these tools walk the user through the selection of specific options, using strategies that are essentially simplified expert systems. But even the more easily navigated of these systems give only specific and limited options, creating a process that is not educational, and a product that is only minimally configurable. Expert systems face significant limitations because their abstract needs assessment processes do not work well with an open-ended and unconstrained process like designing. Lucy Suchman notes that “the structure of the interaction is procedural, constituted by a sequence of actions whose order is partially enforced.” (Suchman 99) But tools should acknowledge that people learn by observing themselves: “the sequences of operations and procedures we execute; the clues we observe and the rules we follow; or the values, strategies, and assumptions that make up our ‘theories’ of action.” (Schon 25)

At the other end of the spectrum, there are numerous modeling environments that allow everyday users to design a home. But modeling software is not something that’s inherently approachable - most people do not know how to represent their ideas spatially, the way an experienced designer does.

The fundamental issue with both configurators and modeling environments is that of representation. Problems emerge because these systems fail to describe relationships or pieces of information that are not within the representation, but still shape the related

products or processes. Computational representations have this limitation in common with other methods of description, like architectural drawings or model-making; in fact, in working with any representation, the complexity of the actual design product is stripped away. But just as architects may move between sketches and models to get a sense of the thing they are creating, and to develop their expertise, this research aims to discover whether or not a computer-based interface can accomplish the same thing through an interactive, multi-representational approach. Towards this end, this research has endeavored to both clarify the parameters of architectural decision-making logic, and to provide constructive, iterative tasks that span multiple distinct representations and facilitate a sort of “learning by doing” for consumers.¹ The fundamental argument is that the establishment of a hands-on, educational design environment is a critical aspect of consumer-based product design systems.

To engage people, this research proposes a direction that is quite different: utilizing search algorithms to enable participative designing. The concepts discussed herein are meant to address the specific and significant problem of spiritless residential architecture, by proposing and evaluating a system for the specification of choice, in the context of residential architecture. The study is constrained to a simplified context: a middle-income, US-based, multi-family residential development. This context is particularly interesting because of the diverse consumer profiles and the current demands being placed on the industry. As spatial functions and user needs evolve, builders have made increasing efforts to efficiently prefabricate flexible living spaces that are in tune with social context and also adaptable. A decline in the availability of skilled labor and the increased complexity of housing systems has increased the expense of field labor and provided further impetus within the industry to find a new way to build. To promote this effort, the software guidelines developed here are intended to support a new industrial process for architects, developers, and consumers.

Ultimately, this study is founded on the notion that consumer-oriented design systems must support the conceptual exploration and discovery of places, and that they can do so by utilizing the structure of search engines. But this new search must enable the

act of designing, without overly constraining it. To facilitate better conceptualization than that supported by current tools, people need to be able to learn what they want while finding it. Towards this end, search engine tools that have traditionally been tied only to the process of discovery within specific formats, have the potential for expansion into the domain of educational design tools that can attach to multiple representations. The search will be centered upon queries that are spatial and/or diagrammatic, and that may be mapped into a component-based modeling system like Autodesk Revit. Rather than speculate about the effectiveness of any single type of search, the work explores the potential of various types of interfaces for constructive searches, and proposes a common index structure that each of the interfaces could be built upon. The specific bases for the conceptualization interfaces are: natural language, metaphorical role-playing, diagrammatic languages, and flows/sequences.

The primary deliverable of this work is the evaluation of prototypical tools intended to support the conceptualization of spatial configurations. There are two types of tools studied: conceptual querying interfaces and a component-based modeling system.² Evaluation is done through user studies with volunteers involving constructive, paper-based exercises; the results of these evaluations are used to design a system prototype that integrates the different interfaces. Figure 1.3 shows various tags that were used in the exercises. Additionally, the integration of this system's design



Figure 1.3 Tags for User Exercises
Various paper word-tags. These were incorporated in various design exercises

formats with other technologies is explored for both designers and consumers, and speculation is made about the potential effectiveness of these types of systems in improving the sustainability and efficiency of private spaces, and in the opening of service channels between designers, manufacturers, and home consumers.

Apart from describing the mechanics of the system, this work endeavors to create a design context that can improve residential home configurations, in important and diverse ways - primarily through the generation of more generalized, but also individualized, custom designs. But also, indirectly, by improving each consumer's understanding of space and how they exist within it. As Kevin Lynch notes, users derive benefits from a more rich understanding of their environment.³ (Lynch, 111)

The work shown here represents only the first steps towards the generation of a system for consumer-based residential design - there are many details that still need to be worked out. As such, the prototypes, evaluations, and implementations are of reduced complexity. And ultimately, this work describes only one approach in an area of inquiry where many others may be taken. But if we can engineer search tools that work like those prototyped here, we may provide the foundation for a more efficient industrial process, by first involving consumers and designers in spatial layouts and space definitions, and secondly by associating these layouts with autonomous and relatable prefabricated parts.

1. The provision of these tasks is made in support of the idea that constructive processes are inherently educational. I will look in more detail at various sources for learning by doing: Seymour Papert - *Mindstorms*, Donald Schon, and Piaget.
2. Component systems are directly searchable because they can describe each of their parts inherently. Further, in the *Society of Mind* Marvin Minsky has made assertions about how humans store and relate concepts about functionally autonomous parts, detailed further in his forthcoming book "The Emotion Machine". These various parts will therefore be the basis for the conceptualization tools.
3. The schematic partitioning of space is certainly a much older concept, but Lynch is notable for the specific characterizations made through field observation.

2 *Design by Searching*

2.1 *Design Tools*

As a category of software, current non-expert design tools quickly highlight the most fundamental constraint of any computational system: the limited descriptiveness of the representations the system uses. We arrive at this problem so quickly because non-expert design tools, as a rule, must inject educational, or at a minimum, *supportive* criticism into whatever representation of choice they happen to be working with, to help guide the user through the process of designing. The criticism structures are tied to and constrained by those representations. There are two common problems that rise from this: first, the tools cannot scale beyond the limits of the representation. And secondly, the user's learning support systems are tied to the specific representation of the tool they happen to be using, rather than the conceptual design domain within which the tool is attempting to facilitate the discovery.

Designing in a complicated domain like architecture requires knowledge about forms, structure, lighting, materials, and so on. Moreover, the process is a scalable one; often architects will ignore some details of the problem to address other design concerns, and are constantly shifting their area of focus. For example, an architect may be very concerned with the organization of spaces around a central hall, and may do some design sketches of the plan configuration without trying to work out the details of what the walls and roof will be made of. These other things would be "saved for later". At first glance, the architect may appear to be simultaneously managing all sorts of different design considerations, and selecting between them with computer-like precision until a final design is resolved, complete in every detail. But research has shown that the architect's process is far less controlled, and involves the constant discovery, and re-discovery of problems and solutions within the context of a design.¹ The professional act of designing is far more of an improvisational act, with some structure provided by project constraints but with tacit knowledge, and reflective learning constantly at play.

The idea of constant discovery is an important one because it highlights one of the most critical functions of an architect, and that is to find and frame problems as a way to steer through the immense potentiality of the physical solution space. Architectural design is like a search for good solutions. But it's a search for the unknown, not a search where you know everything ahead of time. It's like trying to find your ideal mate, not like looking for a penny on the sidewalk. You might know a few things that the person should be or have, but your ideal mate is likely a complex fit - and probably not very easy to describe completely.

The first efforts to make software tools for novice users attempted to engineer professional knowledge that guided people through the architectural design process. These solutions could be described as basic expert systems, where a linear narrative of "value" questions led to a design recommendation. An early example was the ARCHIT expert architectural design system from the 70s (Figure 2.1), which was criticized for its limitations. (Negroponte) The failure of these systems quickly highlighted the problem with their approach: improvisational, tacit designing, which all good architectural design consists of, is not easy to encode as a script. First, the design process is not linear; it may be circular, tangential, or even random. And abstract realms of knowledge, like aesthetics, may be a source of

```

WHICH TYPE OF SINK WOULD YOU PREFER:
1) A SINK WITHOUT COUNTER SPACE?
2) A SINK WITH COUNTER SPACE ON ONE SIDE ONLY?
3) A SINK WITH COUNTER SPACE ON BOTH SIDES?
23
DO YOU WANT EACH OF YOUR CHILDREN TO HAVE SEPARATE LAVATORY FACILITIES?
? No
WHAT KIND OF SINKS DO YOU WANT FOR YOUR CHILDREN:
1) SINKS WITHOUT COUNTER SPACE?
2) SINKS WITH COUNTER SPACE ON ONE SIDE ONLY?
3) SINKS WITH COUNTER SPACE ON BOTH SIDES?
23
HOW MANY GUEST LAVATORIES DO YOU WANT?
21
IN A GUEST LAVATORY, WHAT KIND OF SINK DO YOU WANT?
1) THE TYPE WITHOUT COUNTER SPACE?
2) THE TYPE WITH COUNTER SPACE ON ONE SIDE ONLY?
3) THE TYPE WITH COUNTER SPACE ON BOTH SIDES?
21
DO YOU PREFER:
1) SHOWERS
2) BATHS
3) EITHER OR NO PREFERENCE
22
# COMPONENT SPACE DIMENSIONS
A B C D E F
15 MASTER SINK 1.25 1.40 1.25 1.25 2.33 0.00
16 KIDN SINK 1.25 1.40 0.00 1.25 2.33 0.00
17 KIDP SINK 1.25 1.40 0.00 1.25 2.33 0.00
18 GUEST SINK 0.00 1.40 0.00 1.25 2.33 0.00

I A I B I G I
|----|----|----| -
|CNTR |SINK |CNTR| 0
|----|----|----| -
.....ACCESS..... X
.....

```

Figure 2.1 ARCHIT Screenshot
Example of an early, narrative expert system interface for architectural design

decisions in the design realm. Flat, linear systems simply didn't allow people to relate, in a rich way, their own knowledge to the problem, or to discover their own solution within the problem space.

This incompatibility with abstract design sources points to an important criticism of design engines: the flat-out inability to handle or encourage metaphorical and associative thought. This is something that people are really good at, and something that human designers are constantly doing. For example, the architect may borrow ideas or fragments of structure from other projects to arrive at their own solution. Or the entire design process may be metaphorical: the architect may, for example, try to evoke the nautical sense of a sail with a sweeping roof design. These ideas are context, time and space specific; they may even be discovered by accident. For non-experts, this type of associative virtuosity should also be enabled and supported, for it is the most basic of conceptual structures, and organizes both the design inquiry and the designer's thinking.² In the realm of associative thought, everyone is an expert simply by being alive, aware and human.

More recent consumer design tools have emerged that associate various design interfaces with software critics. (Figure 2.2) These critics are programs that evaluate designs according to a specific, encoded rule-set.³ For example, R. Williams (MIT House_n) explored methods for training software programs how to criticize a user's designs by comparing those designs to pre-evaluated examples. (Williams) These tools represent a significant improvement over linear scripts because they are iterative in nature and allow the user to make (and learn from) mistakes, by highlighting those mistakes in some way. (Figure 2.3) This makes interactivity inherent in the process and allows for more self-direction, depending on the representation used. An everyday user could conceivably create a simple floor plan by placing pre-arranged room configurations, and upon completion, a critic could be invoked to check out specific properties of the design - for example, to make sure the rooms all connect up, or that the toilet is connected to a water-wall. In application, this works better for someone who is an expert already, because having the critic discover more than a few of errors creates a dizzying scene for a novice, where the logic behind the critic's

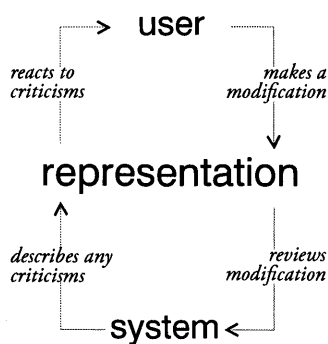


Figure 2.2 Design Critic Cycle
The iterative cycle of an embedded critic sub-routine

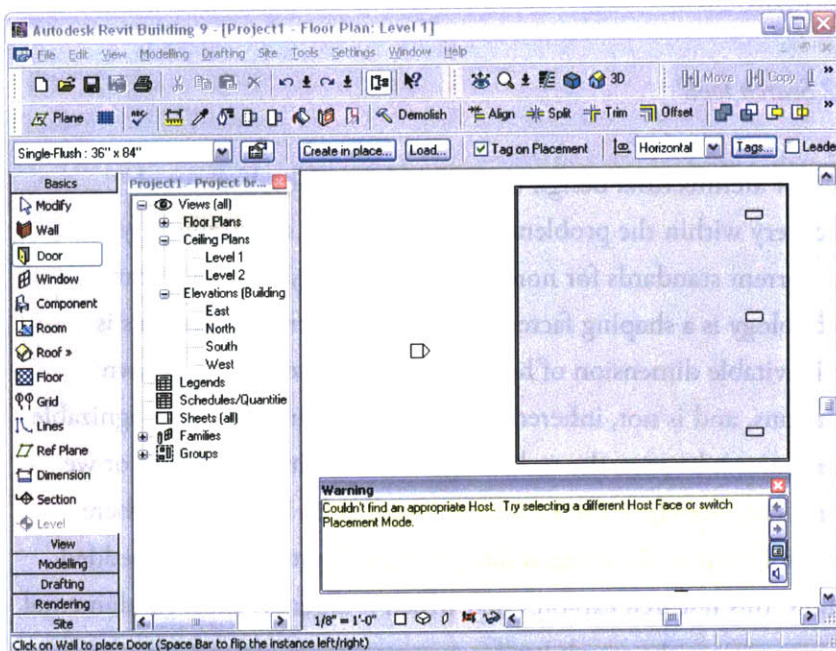


Figure 2.3 Design Criticism in Revit
 Here, the user has attempted to place a door in the open floor. note the prompt

suggestions may not be clear, or simply where the number of issues is overwhelming. Another issue is that of the encoded rule-set used for criticism: critics must infer facts from the specific details of a representation and then evaluate these inferences against rules to discover correctness. The non-expert knows nothing about these rules, which, of course, is why they were encoded in the first place. But this is also their biggest problem: they represent a black-box of hidden mechanisms where confusing outputs frustrate, complicate, or terminate the design process.

For the everyday user, there's also a problem of distraction. Where the representation may have allowed them to more freely explore potential solutions, the criticism quickly pulled them out of their conceptual problem space. At the lowest level, this is because the criticism is tied directly to the system's representation, and cannot scale to fit different conceptual representations the designer may have been holding in their own mind. So quite simply, things may be surprising or make no sense. As learners, adults are self-directed (Lieb): they must be able to perceive the immediate or long-term benefit of working through a problem, and they must be able to relate the design to their own experiences on a conceptual level. This relation must be allowed to persist throughout the act of designing, because it's the fundamental mechanism for seeking out and selecting

solutions.

2.2 Search tools

For architectural design tools, the ability for self-directed discovery within the problem domain is highly constrained by the current standards for non-expert technology. The fact that technology is a shaping factor upon the experience of humans is an inevitable dimension of how people interface with their own creations, and is not, inherently, a problem. But when a recognizable need cannot be met, the technology must be either retooled, or we must look elsewhere for a solution. And so, in looking elsewhere for a solution to the integral flaws of expert systems and embedded critics, this research explores that which is arguably most fundamental to computing: the search itself.

Computing technology is structured around, and perhaps informs, the popular conception that there is some sort of divide between searching and learning – that you must first find some resource, and then, only once you have found it, begin to learn from it. But in everyday life we often observe the contrary: that the act of searching, especially through reflective self-evaluation, is inherently an educational process and the learning is related specifically to that which is being searched for. Consider again the example of finding an ideal mate; many people conceive of the process of dating as a way “to find someone” during which people learn a lot about themselves and what they value in a spouse. The same type of dynamic, reflective searching is particularly true in the context of design.

The rise of the internet and the democratization of information access into more horizontal network structures have naturally led to a parallel rise of web-based search engines that allow people to navigate the immense search space of the internet. (Figure 2.4) In fact, as the primary window into cyberspace, these search tools are in a position of fundamental, central importance: just as Vannevar Bush speculated in the late 40s, it was managing the access and retrieval of an unprecedented amount of information that was key. Prior to the World Wide Web, early search tools like Archie and Gopher allowed people to search file names but not contents. With the

1990	Archie
1991	Gopher
1993	Excite
1994	Yahoo! Directory WebCrawler Lycos Infoseek AltaVista
1998	Open Directory Project Google

Figure 2.4 Early Web Search Tools
Original tools like Archie and Gopher were essentially list searches

advent of the Web in 1993, search tools that could parse full page contents were already emerging. With the bar raised to full content searches, indexing systems were developed to manage the immense and ever-expanding search space of the Web, and to make searches more efficient in real-time. A pioneering researcher in this aspect of searching was Gerard Salton, whose team developed many algorithms for retrieving information more quickly, including the fundamental concepts of document relevancy by word frequency (Figure 2.5), and the automated indexing of documents to make them simpler to parse. To accomplish this, various crawlers (also known as spider) programs emerged, their purpose was to automatically navigate through sites, following links and writing what they see into a central database. This central database thus becomes the collective index. Of course, the index is never complete and never up-to-date, in the context of the Web it creates the illusion that the entire Web is searchable when in fact it is not. The phenomena of unreachable, dynamically generated content that escapes the notice of crawlers became known as the Invisible Web. (Bergman) In fact, the improved search functionality of sorting documents by popularity, which helped Google emerge as the leading search tool from 2000 to present day (2007), actually increases the tendency to overlook many resources. (Bergman)

In discovering the Web, one quickly learns that there is a standard format and methodology for searching, and for discovery: a text box that you can put text in, with a button next to it. You type something in, click the button, and up comes a listing of available and matching resources: behind the scenes, the listing may be ranked, cached, intricately sorted, maybe even paid for, but to the user the listing is always conceptually the same: an ordered set of results. If the results aren't quite right, you can use some different words and try again, until you find what you're looking for. All of this is really just a roundabout way of saying that most Web users know how to search interactively, and most Web users have a mental model for "query", "search submission", and "browse results". This process is fundamental, inseparable from browsing, maybe frustrating at times, maybe educational at others.

Therefore, if we conceive of a search operation as a new

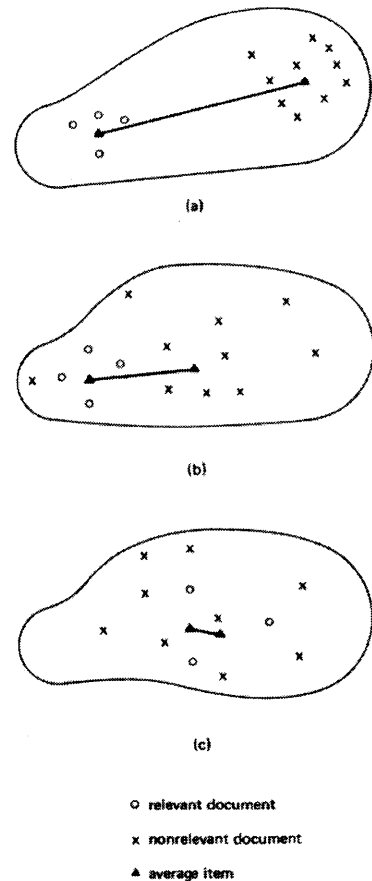
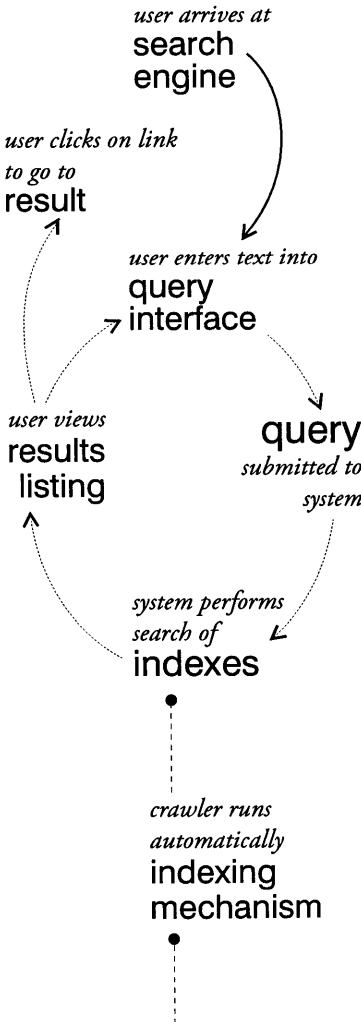


Figure 2.5 SMART Relevance Feedback
 From *Introduction to Modern Information Retrieval* by Gerard Salton, 1983

model for a software-based design process, we have the advantage of a democratically accessible and standardized set of primitive components, which we may then overload to support designing.

2.3 Overloading a search

If searching is a fundamental part of designing, then could the computational search engine be utilized as a structural foundation for a design system? This research supports the development of search tools to help everyday people conceptualize and design physical spaces, recognizing that design-based searching should be a more constructive process, and that some new avenues will need to be explored.⁴ So, in what ways does the standard conception and implementation of searches limit the learning of the human who is searching? Could a search be expanded to allow for the user-defined specification of non-textual, spatial-diagrammatic queries, and could the construction of these queries become an inherently educational process? Designers must be able to search for solutions by iteratively working through a problem to construct a solution. By examining current search technologies in terms of their constituent parts, we may illuminate potential areas for further study, or for injection of new constructive processes.



Search tools have three primary components: an indexing mechanism that summarizes the search space (typically crawlers or directories), the querying interface where users enter a search term, and the results listing. (Figure 2.6) Web-based search tools are generally indexed for manageability and performance. Most search tools are text-based, such that the query, the results listing, and the searching functions are dealing with natural language handling, parsing, and matching. Some newer search tools integrate the conceptual or semantic organization of textual terms in a map-like image, but these are still fundamentally text searches. Recent improvements in computer vision algorithms have lead to the development of visual/image search functionality; for example, e-commerce sites that allow consumers to search for products by comparing items to a source image. The more advanced of these visual searches even allows users to rank what is most important to

Figure 2.6 Web Search Engine
 Typical structure of an indexed search, crawler runs on its own, asynchronously

match between the images, be it color, size, or shape.⁵ For all types of search, the results display will always be sorted: there is an expectation among search users that the top results will be the best by some measure; this measure may be relevance, popularity, influence, or something else. (Fensel, et.al. 18) The query is generally submitted through a simple dialog that then launches the results listing in a discrete page or interface. This allows the search dialog to stay up, and facilitates a smoothly interactive process of query refinement.

Because searching is fundamental, every computer user has some sort of understanding of how a search works. And this is exactly the reason why search functions are such a compelling source for building more powerful design tools: they are fundamental, widespread, and well-known to virtually all users. What better foundation for enabling the exploration of designs, for a large community of diverse users, than search engines? Everyone who uses a computer has a conceptual model of what a search engine does, and what its basic parts are.

2.3.1 The Query

On computers, the search term, or query, is almost without exception a text string. Various rules may be encoded into it, like quotes to match phrases or meta-rules like the “define:” directive Google provides.⁶ But knowledge of these advanced features isn’t necessary, and ultimately, the standard query is still flat, abstract text. Newer interfaces have begun to integrate visual organization into the query, in an effort to create a process that is more intuitive. For example, some web sites use a semantic diagramming structure; the idea behind this visual representation is that it helps people navigate the search space by grouping concepts together and relating the groups into a kind of primitive organizational structure.⁷ The search query is still plain text, even though the representation is more visual in nature. But could the process of specifying the query be something more constructive, something that allows users to assemble visual information? Could it be something along the lines of drawing or composing an image, and what would be the benefit of a visual query as opposed to a textual one, in the context of architecture?

The motivation for moving towards constructive queries is to improve the learning supported by the search process in general, and to improve the mapping of search concepts to discovered architectural solutions, in particular. Let's start with the learning part, and Jean Piaget.

Over the course of his life-long research into the cognitive development of children, Piaget closely related the mental development of children to different types of interaction with the environment. As children grow through various stages of knowing, their understanding of the world evolves from simple conceptual structures related to movement and objects, to the development of motor skills, followed by logical thinking and ultimately abstract reasoning. (Piaget) While there is no clear division between these stages of development, they are all fundamentally about iterative cycles of doing, and they indicate that knowledge is fundamentally constructed internally by the child, through processes of reflection. This new perspective lead Piaget to a set of theories about learning by doing, known as Constructivism.

Others would pick up on this line of inquiry, as did Seymour Papert, who had worked with Piaget in the 1960s. In 1980, Papert published *Mindstorms* and formally described his concept of Constructionism, which draws in part from Constructivism, but further identifies the process of building things as an intensive and primarily educational mechanism. From his observations, Papert contrasts basic mathematical knowledge developed through traditional methods with Constructionist methods of instruction, finding that the latter lead to concepts that were more relatable to various types of tasks, and also more significant to the children. (Papert, 53) Further, he describes an important point: that the knowledge built through making things can be developed just as effectively by various types of learners, even kids that were previously “mathophobic”, because the process of construction enabled them to relate things that were meaningful to them as individuals. (Papert, 63)



Figure 2.7 Children with Turtle
Learning with Logo, from *Mindstorms*
by Seymour Papert

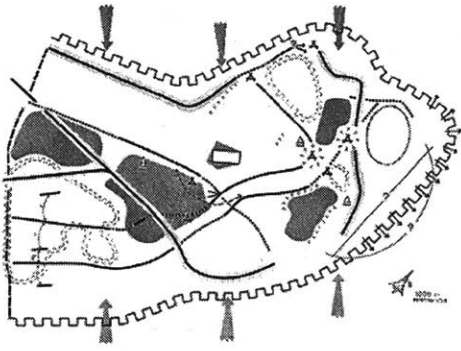
To help people learn about architecture, this work tests the viability and accessibility of spatial or conceptual images that are constructed, iteratively, by the user and submitted as a search query.

The fundamental challenges with this are first creating a way to construct searches that people are comfortable with, and secondly determining the level of descriptive capacity that people really want to have access to, and finally determining whether that level of description can be matched effectively with various computational representations of architecture.

This research is aimed at tools for the potential home owner, and therefore, adults. Adults really aren't as efficient learners as children are, because they're influenced by experiences and expectations. (Minsky 1, 92) There are innumerable reasons this may be the case, but experts in the field generally agree that it has something to do with the following properties of adults. First, they have more demotivating roadblocks: they may be short on time, or may have other responsibilities or concerns that prevent them from digging into the task. And they are far more self-directed: adults want to be able to pursue a problem in the way that is most comfortable to them. (Lieb) Finally adults must be able to perceive the immediate benefit of something before committing the time to learn about it: they will have more resistance to learning new things at the onset if the interest isn't there inherently. In the context of a more visual search, this means that the task of image-making, which would probably be new to most users, is potentially daunting. In fact, upon realizing that almost all Web searches are text searches, one might speculate that the general consensus among search developers is that people simply can't or don't want to do searches by image-making.

To get a sense of what may or may not be doable, different types of conceptual querying interfaces that related searchable concepts differently were evaluated. Rather than speculate about what sort of searching interface might be the best one for our purposes, this sort of evaluation gives us a better foundation for future work and opens up the possibility of using different query interfaces, or different combinations of queries, for individual users. This research evaluates the effectiveness of four querying interfaces: text-based, role-playing metaphor, activity-based, and diagrammatic. Chapter 4 will provide a detailed summary of each interface type.

Among those tested, diagrammatic representations are the most difficult to search with, but they're attractive because they have



enough constraint within them to be understood by a computer, and they also make sense to people, as suggested by the work of Marvin Minsky and Kevin Lynch. However, being able to make sense of diagrams and being able to actually create diagrams are two different things; the diagramming process must be approachable. (Lynch 11) Furthermore, because the diagrams are potentially much less constrained than the parameters of the search space, there is greater potential for the user developing unmet expectations in the act of diagramming. In other words, they may think they are specifying something meaningful as a search criterion, where in fact they are not. This will be discussed in greater detail in the following sections.

In his research about how the human mind may structure and retrieve information or knowledge, Minsky speculates that there are likely many different types of structures at play, with different types of functionality to facilitate different types of knowing. Among them are semantic networks, which relate together the various parts of entities or processes. These networks may be assembled into narrative sequences, which might related to simple constructive processes, like what happens when you move a chair, for example. (Minsky, 137) The fundamental idea here is that people understand that which they see in terms of their constituent parts or properties, and that these parts might be relatable, or swappable, to facilitate low-level associative thinking. Low-level associative thinking takes many forms and is evidenced by metaphorical relations; for example, thinking of a slice of pie to describe a pie chart.

In the specific context of spatial configurations, Kevin Lynch's research on how people understand their environments pointed to a common language of parts for these knowledge networks, at least at a high level. Through evaluation, he discovered that people's conception of space could be generalized into a few specific types: paths, districts, edges, landmarks, and nodes – and that these types of things could be related diagrammatically. (Figure 2.8) He extracted these findings through the case study of specific cities, and conducting interviews with inhabitants. The basic types that Lynch describes are the foundation for the specific diagramming language evaluated herein, where districts and edges taken together are simplified into the cell(s) of a grid. Nodes and Landmarks are characterized as particular

Figure 2.8 Lynch's Diagram of Boston
A problem map that summarizes urban wayfinding information
From *The Image of The City*
by Kevin Lynch, 1960

functions or activities granted a specific location. And pathways persist as a way to connect the other primitives together, to create basic assemblies or systems.

Taken together, the work of Minsky and Lynch suggests that simplified diagramming processes may map well to human thought, and specifically human thought about spatial organization. The potential area of difficulty in constructing the diagrams is that making them forces one to think about thinking. Architects for example, seem to love diagrams as a way to organize thoughts about things, this is like a meta-process for them, and inherently reflective. But this type of high-level analysis may be an endeavor that only the experts have the knowledge to pursue. Would beginners even know where to start?

Rather than focus upon just one strategy, this research is exploring the different characteristics of four unique query interfaces. The primary reason for this is to provide users with multiple options for expressing themselves, particularly if one of the interfaces seems difficult or unbeneficial. In addition to the diagramming interface, this work examines an interface that allows users to list out each activity in a routine that happens in their home. Another alternative is tested as well, an interface that allows users to imagine packing up their various possessions, as a way to think about their needs. And finally, text-based checklists are provided as a more standard type of interface. Each of the interfaces is described in detail in Chapter 4.

2.3.2 The Search

As long as there have been computers, people have been thinking about methods for searching. In his influential 1945 article *As We May Think*, (Figure 2.9) Vannevar Bush cited the need for technological advances in searching: “The summation of human experience is being expanded at a prodigious rate, and the means we use for threading through the consequent maze to the momentarily important item is the same as was used in the days of square-rigged ships.” (Bush) Bush was arguing both for the record of data, and for associative, conceptual tools to help people navigate the data.

By 1965, electromagnetic data storage was more widespread.



Figure 2.9 *As We May Think*
By Vannevar Bush, published in
Atlantic Monthly in 1945

Gerard Salton, a researcher in the then-emerging field of computer science, pioneered the era of modern retrieval algorithms with the SMART information retrieval system. His system introduced important indexing concepts like document frequency, term frequency, term discrimination, and relevancy feedback for text-based searches. (Salton)

Salton’s work introduced concepts that are still very much in use today. At around the same time, Ted Nelson began publishing about his concept of “hypertext”, or the embedded tagging of metadata into informational resources to apply structure and accomplish attribution. The World Wide Web would eventually be developed using the idea of hypertext for its fundamental document structure, and the tagging mechanism is what enabled the more efficient and robust indexing mechanisms to be developed. (Figure 2.10)

The work of Salton and Nelson was on the leading edge of search tools that were built to function through indexing. In the context of searches, indexing refers to the retrieval, parsing, and storage of data for improved efficiency. Particularly in the context of the internet it emerged as a necessary element of search design simply because of the size of the potential search space. When you do a search on Google, for example, you’re not searching the internet in a “live” fashion; you’re searching Google’s index of Web resources. Google, like all Web search engines developed from 1993 on, utilizes specialized crawler software to automatically index the Web by navigating through links and parsing the web pages that are encountered. These parsed pages are then stored in Google’s databases, which means that the search can be faster and more reliable in terms of delivery, but never completely up-to-date in terms of the content. And because much of the Web is dynamically generated by server-side programs that even the most advanced crawlers cannot parse, much of the Web becomes un-searchable, invisible.

But of course, this is the reality of the immense problem of searching the Web. Apart from global Web search engines, many web sites have their own searches which read through the internal information that drives the site itself as a way to deliver content.⁸ These searches do not need to automatically generate indexes; the internal information is queried directly.

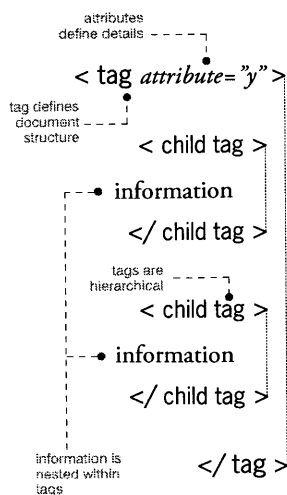


Figure 2.10 Hypertext Example of HTML tagging structure

This research explores functionality that is very much like a Federated search, which searches for results from a specific group of known resource providers. (Figure 2.11) In system described here, the providers are builders or developers that want to offer searchable homes. So the system is situated between the small, site-specific solutions and the global, indexed solutions like Google, and will highlight additional problems and complexities. In order to associate the previously described conceptual query images with various types of architectural representations, we must describe not only a common indexing mechanism, but also a mapping process that translates the conceptual to whatever literal representations we want to be able to search for. The index which is proposed and evaluated is an XML-based record of the conceptual structure. When an image query is constructed and submitted by the user, the XML description is automatically generated. In the search space, resources may have their own XML description which can be compared directly, or they may be mapped to the query in real-time by an intermediary application. However, this system does not require any automated crawlers to facilitate indexing functions, for several reasons.

First, as opposed to the incalculable size of the internet, our search space is much more constrained. Under the Open Source model and as detailed in the subsequent chapter, a centralized conceptual search may be used to query finite, discrete resources that are offered in various formats by specific developers, builders, or

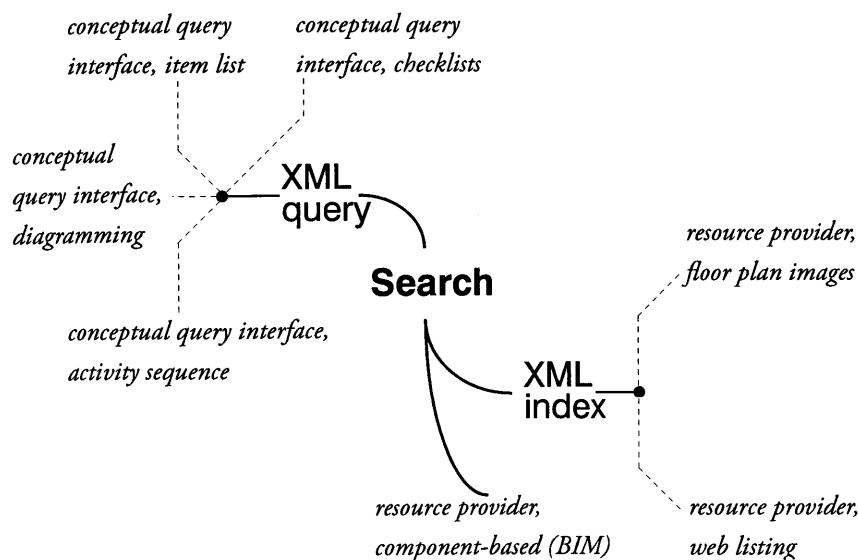


Figure 2.11 Proposed Search Structure
Different query building interfaces to search different types of representations

architects. Each of these resources is accessible to the search interface through mapping functions, or by direct comparison of the resources where the representation allows for this. As such, the XML indexes represent more of a common specification, a meta-representation to use between other representations, and the submission becomes a more complex operation of, for example, mapping conceptual searches to JPEGs.

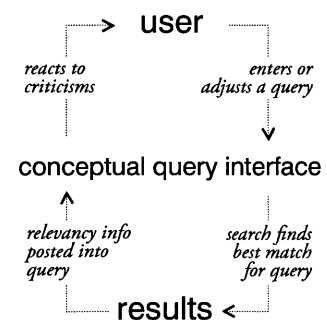
Search submission is most often thought of as a one-way process, but in truth it's far more interactive than that. Using Google as an example: the search term is submitted, and processed against a complex matching algorithm against available resources, where the available resources are the entirety of the Web. Google's result listing allows the user to both reflect upon their query (if the results were unexpected) and to revise and re-submit. So the search process provides criticism and guidance via the relevance measure of the results. In the context of the proposed design tool, let's assume the available resources would be designer-provided floor plans. Each floor plan would further specify a mapping protocol to dictate how the searches' primitive components should map into physical form. It is here, within the submission process, that the essential design criticism may be embedded by describing the relevancy score for each available resource directly within the user's original search image. This enables the relevancy information to provide criticism and guidance to the user inherently, as part of the searching process. The criticism logic is constrained neither to the conceptual nor the literal representations; it exists between them. The criticism is emergent, derived from the relationship between the query and the results themselves.

In his discussion of the emergence of different and late-period styles in music, Edward Said asserts that part of music's structure and purpose is to remind the musician of the specific styles of the time. He notes: "Were this reminder to be simply a repeated *no* or *this will not do*, late style and philosophy would be totally uninteresting and repetitive. There must be a *constructive* element above all, which animates the procedure." (Said) In the context of the proposed design search, this same sentiment is reflected by the criticism process, which may serve not only as a stylistic guide, but as Said further describes, a

set of parameters within which styles can emerge through constructive and iterative processes.

Structurally, the criticism components would be part of the common XML specification. The act of search submission becomes an important part of the iterative cycle of searching by designing, because it allows the user to instantiate the criticism automatically upon invocation. There are a few reasons that the two-way search process, and thus the posting of criticism functions back into the search dialog, may be challenged as unnecessary.

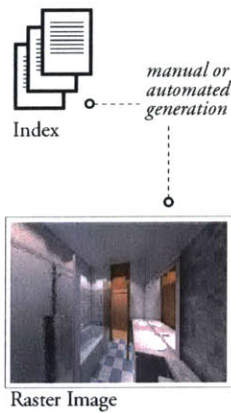
First, as shown in Figure 2.12, there is an inherent criticism process in the standard search where the user may reflect upon the correctness/effectiveness of their query simply by reviewing the result set to see if it contains what they are looking for. But, the problem with a design search is that the user, in general, will not know what they want. So they won't have a good way to evaluate the utility their search image without a little assistance. Secondly, common text searches can do things like highlight words in the results listing, to let people see what words matched up. But along those lines and because the source query is a visual one, the relevance of the results is best described visually as well, and in the conceptual context of the query itself. And because of the wide potential for multiple target representations accessible through the mapping functionality, the query image becomes an important common thread that the user may both construct in and evaluate through. The diagrammatic representation can describe how the results are a good fit, and how they are not, simply by highlighting the portions of the diagram that match. This builds upon the inherent iterative cycle of the search but incorporates new spatially organized relevance and scoring strategies, as a way to help the user learn about design parameters.



2.3.3 The Results

Web Search results are of a common type - HTML. Powerful search tools can automatically parse related formats, like Google does with PDF, DOC, TXT, RTF files, seeking them out the same way that Web content is searched for. But the search matching is still text to text: even in special contexts like the Google image search,

Figure 2.12 Criticism as Part of Search
 Relevancy information provides design guidance to the user.



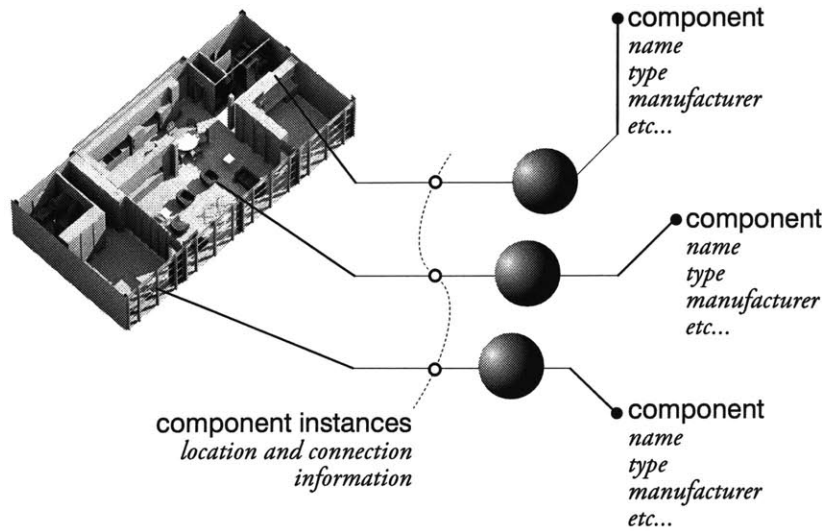
text-based meta-data and text-to-image relational algorithms actually “find” pictures by finding text. ⁹ In the proposed system described here, mapping functions will be explored as a way to relate the search image to any different type of architectural representation: a tagged image, a component-based model, an adjustable vector-based floor plan. This mapping is managed through a common language which, as detailed in the previous chapter, is the XML index. But how would this index get related to various types of representations?

In this sort of scheme, raster images would need to have an XML description generated, and that would be paired with the image as an index for comparison. (Figure 2.13) While this process of indexing could be either human-managed or automated, it’s problematic either way. The development of more advanced computer vision algorithms makes more plausible the idea of using programs to automatically scan and tag raster images, but the level of sophistication to reliably analyze and tag floor plans has not been accomplished. The details of this class of problem in computer vision are outside the scope of this research. And vector compositions and 3D models are problematic in the same way. People could tag these representations themselves using an auxiliary tagging application, perhaps as the designs are being created. However, depending on the complexity of the tags and the number of images to be tagged, this process may or may not be manageable or sustainable.

Component-based representations or models solve these basic problems of searchability simply by allowing for the autonomy of parts, but they also lead to new challenges. Because their representation is made of autonomous entities, the computer-based attribution process becomes straightforward. (Figure 2.14) In the current (2007) world of professional tools, we’re seeing a shift towards design tools that facilitate the construction of component-based reasons, primarily to build a standardized platform for interoperability and smoother workflow. In a way, component-based representations are more intelligent about the physical world simply by virtue of being able to distinguish things. Let us consider, as a case in point, the evolution of AutoDesk’s product offerings.

Autodesk was an early developer of CAD systems, and has continued to be an industry leader in their area of specialty, emerging

Figure 2.13 Image with Auxiliary Index
Different query building interfaces to search different types of representations



as the de facto standard of computer-based drafting tools through their leading product: AutoCAD. While developed around specific functions, processes and commands that are inherent to a computer program, AutoCAD, at the lowest level, was designed in mimicry of traditional pen and paper drafting processes. In AutoCAD you're essentially drawing lines, which may be given simple attributes, in terms of how they should be presented: weight, color, and style for example.

In 2002, Autodesk embarked down a different road with the purchase of Revit Technology Corporation and their new modeling tool. Shortly thereafter, upon the release of *Revit Building*, Autodesk began supporting drawing of an entirely different type: Building Information Modeling (BIM). The main idea behind BIM is that the drawings are no longer made out of lines or vectors, but autonomous components that have specific labels, properties, and functions. The primary motivation behind this type of tool is to streamline the process both for communication between different companies and organizations, and also to simplify the change and revision process. A recent NIST report estimated conservatively that \$15.8 billion was lost annually by the US facilities industry, because of "the highly fragmented nature of the industry, the industry's continued paper-based business practices, a lack of standardization, and inconsistent technology adoption among stakeholders". (Gallagher, 7) Component-based Solutions like BIM, which structure and support information exchange, are a promising solution. But with this

Figure 2.14 Component-Based Format
Object-oriented representations like BIM relate objects with specific functions and properties, rather than lines

autonomy of parts being inherent in the representation comes many other qualities, one of which, I'll argue, is inherent search-ability; thus, accessibility for consumers as well.

Referring back to the diagrammatic query language, BIM systems would be able to map to searches directly: activities relate to components, zones to containers of components, and connections to the hierarchical relation of components within certain containers. In fact, the conceptual search would prioritize these more expressive component representations simply by finding more relevant results there.

The problem inherent to all of these searches, of course, is how well the information they describe or search out may be related to the user's actual sense of what they want. Can the simplified relation of activities in various spaces really give the user a way to describe their architectural preferences? Because these tools are aimed at consumers of all different types and with different motivations, one cannot explore the search potential of BIM, for example, without considering the differentiability of the BIM configurations and how likely the user would be to search them out.

However, as Figure 2.15 suggests, the query could be used for more than just searching for available results. For example, searches could also be submitted directly as a request to one of the resource-providers. This may be particularly useful if the initial search query doesn't lead to any good results, because, as will be discussed, criticism would be tied to the results and would not necessarily be available or adequate. In this case, the user may choose to post their search query to one of the providers, as a design request. This keeps the user, and the user's understanding of their architectural needs (however primitive they may be), at the center of the process, even where the searchability of the representations in the search space is inadequate to map to their search.

In the case of the architect as a resource-provider, this creates the opportunity for a new service-based business channel, one that is both accessible and self-selected by the user.

It is conceivable that the search tool could become a central, organizational structure for the homeowner's ideas about their home. A collection of stored and individualized searches could be parsed

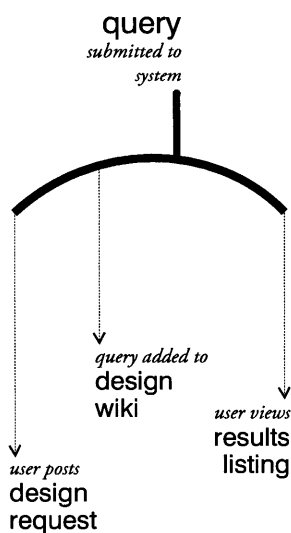


Figure 2.15 Different Query Destinations
Query artifacts could be used for more than searching, they could serve as design requests when posted to a wiki community, for example

into a wiki-style resource full of spatial configurations, with associated comments and other meta-data. Here the search query becomes an artifact, an important node in a collaborative community. Over time, these artifacts could potentially give designers and users insight into the evolution of spatial use, or to improve user understanding of space and space use for better affordability and sustainability.

1. In *Educating the Reflective Practitioner*, Donald Schon provides a detailed

analysis of the design process in a studio-based environment, uncovering a highly reflective, iterative, and at times surprising design process nested in a constant cycle of rediscovery.

2. Piaget's work with children and the development of knowledge suggests that as all knowledge is constructed within the mind, the association of concepts in new ways, or associative thought, is a powerful mechanism for understanding the environment and making new discoveries. Many theorists have built upon these ideas: for example, Marvin Minsky's theories about the structure of the mind accounts for what he describes as *Panalogy*, or the association of representations within or between domains of knowledge
3. Software critics in this context are agent-based simulation systems, which have generated interest over the years for their potential to analyze a representation and to either adjust their behavior or make suggestions to the user based upon this analysis. Agents acting as critics are reactive. The effectiveness of the agent is constrained by both the representation used, and by the programmability of the behavioral logic. In the case of architectural design guidance, critics like this are an incomplete solution at best, because they cannot follow the improvisational logic of the designer.
4. It seems reasonable to assert that even the well-known text search format is an interactive and constructive one, an iterative process of refining the query to get at what you're looking for. The relevance and size of the result-set is what sustains continued exploration.
5. For an example of a commercial visual search, check out <http://www.look.com>
6. A search for "define: word" on Google, where "word" is any word for which the user wants to know the meaning, will return a list of definitions rather than web pages.
7. One example of a semantically organized textual search is Kartoo, available at: <http://www.kartoo.com>
8. Take for example any e-commerce site, like <http://www.amazon.com>
9. As of 2007, Google has implemented a version of the ESP game (<http://www.espgame.org>) to improve their semantic tagging of images in a game-like environment, essentially utilizing humans as a computational resource.

3 The Result Set: Component-Based Representations

3.1 What do Component-Based Representations Describe?

With the emergence and rapid popularization of object oriented, component-based systems comes a new potentiality not only for the architecture, engineering, and construction industries, but for participative design as well. The new potential derives from the fact that the component-based representations are “smarter” than other representations: i.e., they know more about that which they are describing, and from the fact that they provide a standard for sharing descriptions of things.

This work explores the inherent searchability of component-based representations, in terms of their ability to be matched without prior indexing, as a technological facilitator for new business channels and processes. But what exactly do component-based representations describe, and how are they going to help change the way homes get built?

Component systems describe a collection of related objects, where each object is represented by a shape, a specific location and spatial relationship to other objects, and by additional properties like names and product numbers. (Bernstein) When an object is placed and scaled into a drawing, component systems describe the location in terms of coordinates, like most any CAD system does. (Figure 3.1) Objects commonly have an insertion point to which the location coordinates point; rotation and other operations are then made about this point and stored with the specific instance of the object. BIM systems also have the potential to describe location globally and connections at a higher level, between buildings for example.

The shapes are 3-dimensional and defined by closed polygons with a specific number of sides. Certain types of shapes are parametric, depending on the object. For example, a “Copy Machine” object’s definition would not be parametric or scalable; it would simply be a volume enclosure of a specific shape. However, a “Wall” object’s definition would be freely resizable, its length, width, and heights being variable and context-dependant. In addition,

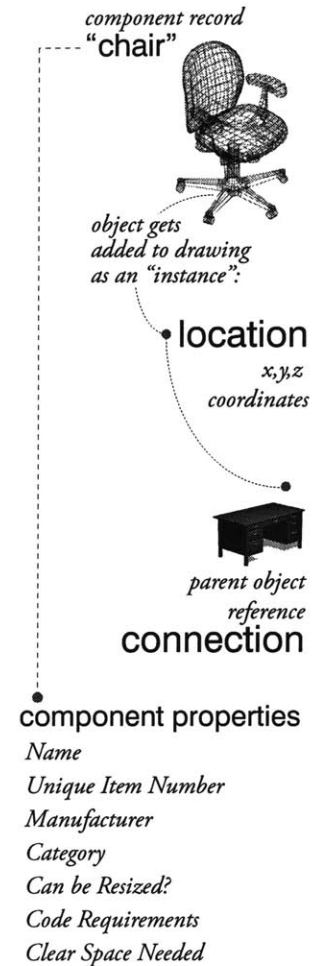


Figure 3.1 Component Details
The Objects in BIM-like drawings are specifically located instances of prototype components

various wall segments could be joined together to create, for example, a room enclosure; these objects would connect into a more complex shape.

A key benefit of component systems is that the shape, however flexible it may be, is automatically associated with other properties that describe the object more completely. The representation is therefore object-oriented, in that each part is conceived as an object with specific properties and functions. The associated properties for each component include categorical types, cost data, manufacturing information, and specific physical requirements, such as what can connect to it, among other things. These connected bits of information make component systems smarter and more computable. For example, automatic criticism functionality can be programmed into the drawing interface, because conflicts between components can be immediately recognized. Consider the placement of a door object into an Autodesk Revit document: this placement is only allowed into certain other objects, a wall in this case.

Thus another part of what makes component systems more computable is that they describe how things should/can connect together. (Bernstein) Just like doors can only go into walls, windows can only go into walls, and so on. In truth, the connections between all of the various parts are what make buildings so complicated; and indeed, Revit's BIM representation becomes useful even as a spreadsheet of what is *in* a building and what is connected to what. Connections in general are not simple, and even in the door example, code requirements and specific clearances (ADA compliance, as an example) need to be enforced and accounted for. But the point is that the representation is smart enough to allow these things to be accounted for.

When various components are connected together in a BIM-type representation, they create an assembly. Assemblies are simply spatial arrangements of connected components. In the context of architecture, assemblies may be made at many different levels, to describe a furniture arrangement, or the connection between a toilet fixture and the plumbing line, and so on. In our evaluations, assemblies describe rooms, and therefore describe specific room functions.

In the realm of buildings, there are all sorts of connections that a component system has to describe. There are service-type connections, as with electrical outlets, lighting, and HVAC, and there are also plumbing connections that need to be made for specific objects, like sinks, showers, and toilets. In addition to that, there are object-level connections made between physical objects, for example the placement of cabinets along a wall means the cabinets need to connect to the wall, and the placement of a table in a room means the table is connected to that room. Furthermore, connections can be made between rooms, where rooms are assemblies of objects that define a specific region. Between rooms, a doorway serves as a connection from assembly to assembly, a relational structure that is fundamentally the same as the object-level and service connections previously described.

Service, object, and room connections comprise most of the connections that need to happen in the typical floor plan, (Figure 3.2) but it is easy to see how the representation might need to scale, to consider for example the connection of a building to its site or a neighboring building. ¹ We will return to the issue of connections in a moment, in describing the specifics of our test implementation.

3.2 Component Systems versus Traditional CAD Representations

The autonomy and identity of parts within component systems makes them more searchable than traditional drafting software packages. Consider as an example, AutoCAD: a popular drafting software published, like Revit, by Autodesk. This traditional package was quite successful when released and is now widely implemented and utilized. AutoCAD's success was quite natural, because it has always functioned well with the technology of the time and improved workflows when compared to paper drafting methods. (Bernstein, 8) AutoCAD was also successful because it preserved, even if only on a conceptual level, the basic production process associated with paper-based architectural drafting. The metaphorical foundation for traditional CAD tools has always been manual drafting: they allow people to construct things out of lines of one sort of another, just as was done with a straightedge. But being an assemblage of lines, the

- **object into room**
where room is a "container" component
- **room to room**
connection is a threshold, and associated with a door object
- **service**
Plumbing connections made to fixtures...
Electrical connections made to outlets...
HVAC ducts, connections made to systems and vents...
...

Figure 3.2 Common Connections
Connections are generally made for things, rooms, or utilities

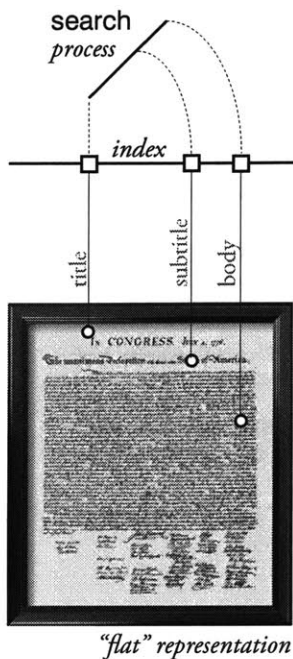


Figure 3.3 Index of Flat Representation
Important aspects of the document are re-represented into a computable index

resultant representation may be very meaningful to a person, but not to a computer.

This is because there is nothing that can be differentiated about the primitives – the system doesn't really know if a line represents a wall or the edge of a counter. Component systems solve this problem by making all primitive drawing elements into objects with names, properties, and purposes. These objects in turn provide the system with an indexing structure that enables it to find and analyze objects directly.

An index, described generally, is something that guides, or points out information for reference purposes. In terms of searching, consider for a moment that indexes and representations are basically the same thing: relative to any physical reality, all representations are indexes, in that they highlight specific dimensions of that physical reality, deemed important simply by the fact that they're represented. If a representational index is too "dumb" to be computable by a machine, automation processes will require that representational index to be indexed, as well. So, indexes of indexes emerge as a sort of fix for the limited dimensionality of a representation. (Figure 3.3)

The need for indexing has arguably been around for as long as any information has been recorded at all, as a way to facilitate information retrieval (IR). (Metcalf 4) But the earliest definitive indexes emerged with the advent of printing processes and the parallel explosion of information in paper formats. Published documents began to have preliminaries attached to the beginning and ending, which summarized the information contained in the document via contents pages and sometimes, an index that pointed out where specific information could be found within (Metcalf, 15) This had become undoubtedly essential due simply to the staggering increase in the volume of information within books but also between books, as evidenced by the parallel emergence and standardization of library indexing of volumes. So there are indexes *of* books, and indexes *within* books.

Of course, modern computation marked another explosion of information, and of more than one type of index. Just as the commoditization of personal computers and the growth of the internet lead to a staggering increase in the number of documents

that would need to be traditionally indexed, so too did documents upon which computer applications automatically operate need to be indexed. One of the best examples of this is actually the programs that people have created for compilation into executable code. (Figure 3.4) All programming languages have indexing inherent within their structure, designed to enable a machine parser / compiler to understand various sections of a program, and to differentiate data, for example.

Where earlier programming languages were procedural or instructional in structure, they have steadily fallen out of widespread use in favor of object-oriented programming (OOP) languages, because procedural languages like C became more cumbersome for people to manage as coding projects became large and complicated.² The reason for this is quite simple, actually: while both procedural and OOP languages are well-indexed for computers, the indexing structure of OOP is more natural for people: OOP is rational to both computers and humans.³ This computer-and-human legibility highlights an interesting dimension of *making* things that are actually computable on computers: unlike a flat representation like a line drawing where only the person needs to understand it, or machine code where only computers need to understand it, computable representations must be readable by both.

Component systems emerged from the application of OOP principles into the process of describing buildings. Realizing that drawings created in mimicry of drafting could not be truly computable, engineers made the data structure for the drawings object-oriented, which is an inherent index for the computer, just like OOP languages. In terms of the architectural, engineering, and construction industry, this helps each of the key players move towards a singular, more descriptive representation for sharing all aspects of a project, and ideally for making all aspects of a project searchable.

This actually marks a simplification in the realm of architectural representation, because it eliminates the need for secondary indexing of the representation and its contents. For example, a professional draftsman has to label all of the objects in a traditional CAD document to make it readable for others: he starts with a basic index. The index would be created manually, either within the CAD

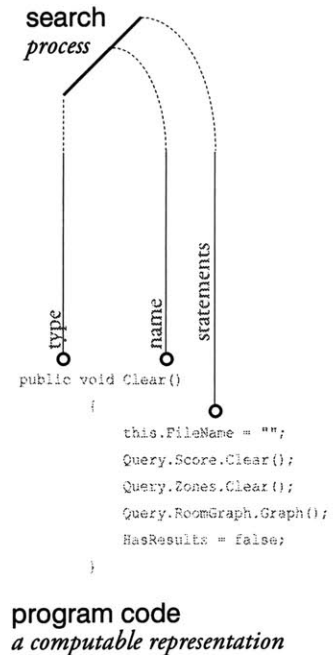
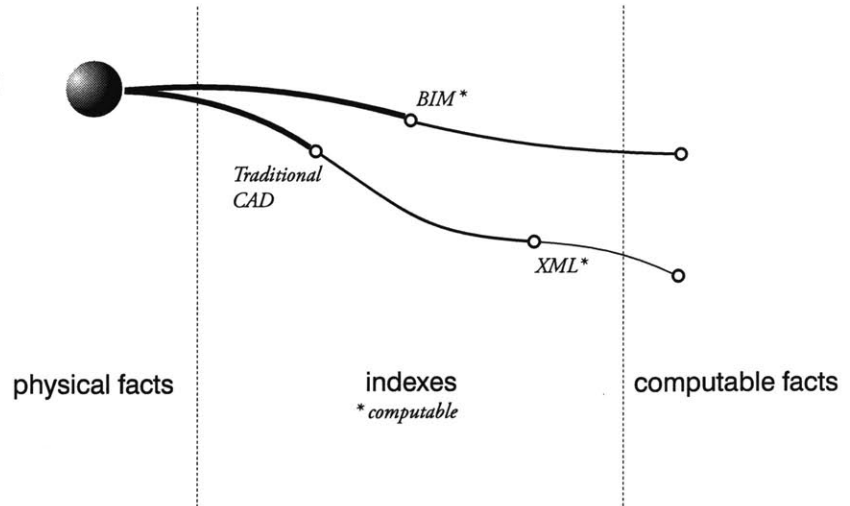


Figure 3.4 Computable Representation
Machine readable representations like the sample program code shown here are their own index

Figure 3.5 BIM Versus Traditional CAD
 BIM is inherently computable, whereas traditional CAD documents must be indexed



document or in another application altogether, by visually scanning the drawing for all instances of all relevant objects, tagging them with a number or symbol, and then translating that number or symbol into a chart that describes the objects. (Figure 3.5) Then when the drawings get printed up, people in the field can make sense of all of the lines by referring to the prepared index or key. If you wanted to program a similar but automated computer search of the drawing, you would have to do the same thing, either by tagging it visually using advanced computer vision functionality or by associating an XML-based index.⁴ Thus, in the traditional model of CAD, there are, in fact, two indexes between the searcher and the physical world: the physical objects are indexed into line drawings, and the line drawings in turn are indexed by some sort of tagging. In the object-oriented realm of component systems, a layer of abstraction is removed because object properties are already described as soon as the object is drawn, so that no secondary indexing is necessary.⁵ (Figure 3.5) Component systems relate directly to the physical; their representations are an index of the built world: descriptive, readable, and just as importantly, searchable.

3.3 Component System Prototype

To test out the inherent legibility and searchability of component representations as a search result, a simplified component system was developed.⁶ Additionally, a paper-based exercise was developed to evaluate the effectiveness of the system through sessions with

volunteer subjects. The prototype system, detailed in Chapter 6 and summarized here, is concerned with how components may be related together into configurations.

The exercise, in turn, has been developed to analyze how these components may be organized into searchable chunks that are meaningful and approachable for everyday users. Component-based representations may be inherently searchable and offer the promise of streamlining the building industry; indeed, of democratizing design, but the description of a building is still a very complex problem. The exercise is one approach to simplifying things, the limitations of which will be discussed shortly.

Figure 3.7 (following page) shows diagrammatically how the component system will function as part of a larger design search.

3.3.1 System Overview

The component system for this study was built within a relational database, further detailed in Chapter 6. On the structural level, each of the components, and thus the assemblies themselves are connected together via an edge-based connection specification. (Figure 3.6) The connection is made between faces of rectilinear objects, where a sub-region of the edge is designated as the connectible portion, and each component can specify not only what types of other components it can connect to, but also how far away it can be (range) and how much of its interface needs to overlap with the connecting interface (fit). Both range and fit can be set as zero: a range of zero indicates that the component must be adjacent, and a fit of zero means that the connection can be made to a point on the face of the component. Predictably, the interface between rooms, being doors in walls, have

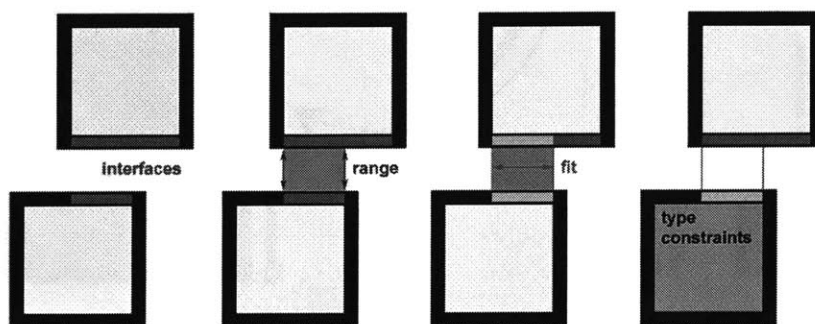


Figure 3.6 Connection Details
A simplified model for relating the components together

Complete System Overview

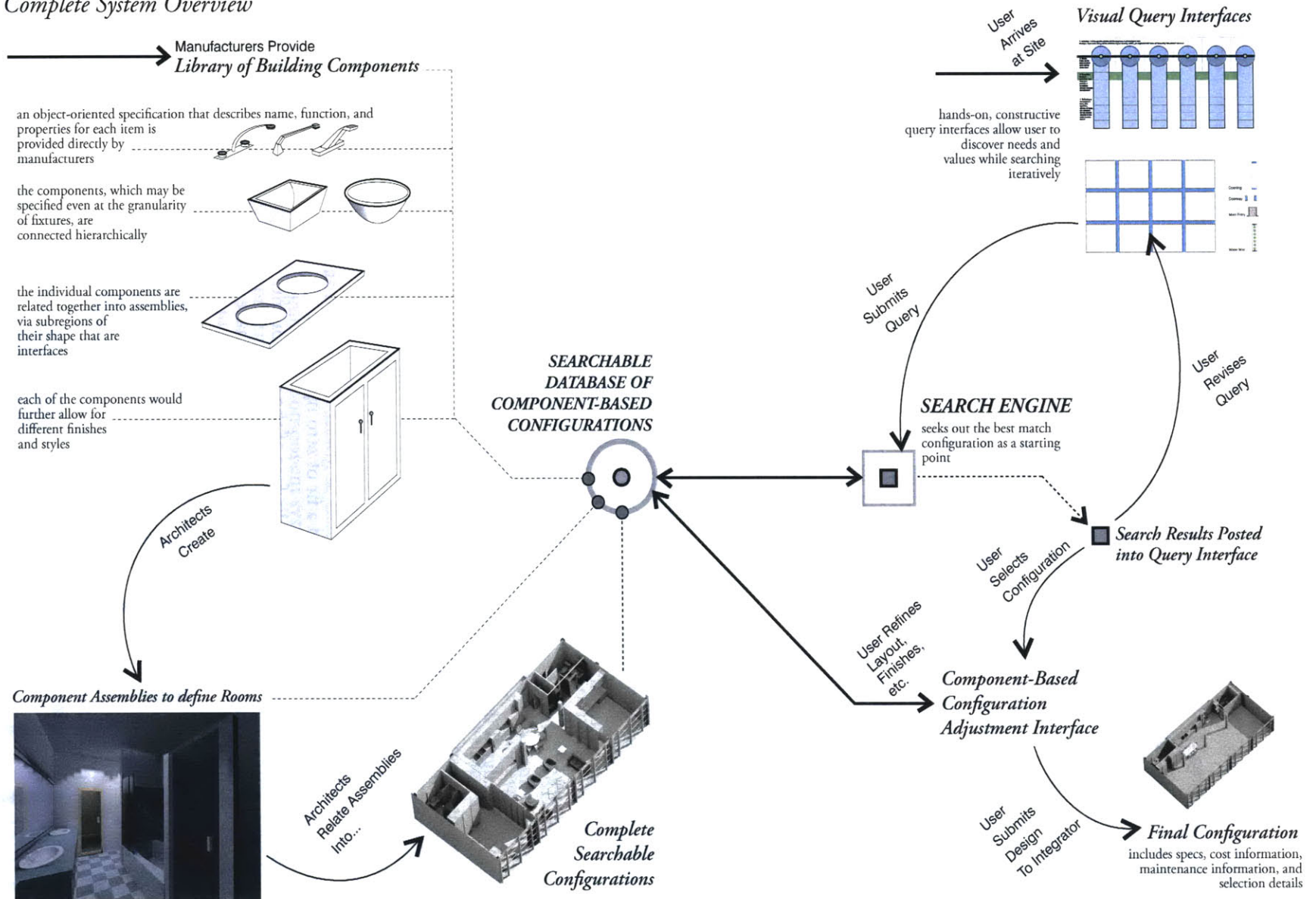


Figure 3.7
Complete System Overview
Diagram of the component-based
participative system including search
functionality and key players

a range of zero; while the placement of a table in a room container would have a fit of zero, allowing it to be rotated and/or moved freely.

The prototype system database was loaded with pre-defined assemblies of components that represent specific room layouts. 145 specific assemblies were designed by a team of architects, making thousands of different layouts possible within the system.⁷ Within the collection of assemblies, each type of room has multiple layout options available; these layouts were constrained to the context of a multi-family apartment interior to simplify the scope of the analysis.⁸ Because the search space for the component system is thus constrained, there are important design parameters that go unrepresented, particularly in terms of site information, and solar orientation of the unit. By focusing on interior fit-out, these significant architectural problems are not addressed. But it goes without saying that any true implementation would need to consider these aspects of designing.

The assemblies for this prototype encapsulate expert knowledge about how individual rooms should be defined, and through both the alignment of doorway connections, and the alignment of zones, knowledge about how the entire plan can be defined as well. As such, much of the architectural decision-making is already done. This raises important questions about how much designing is actually possible with so many decisions already made.

And beyond that, how much choice do people demonstrate the ability to manage? In terms of the user's specific decision-making process, the following exercise hopes to tease out, through the multi-representational library of assemblies, what types of specific representations are most helpful for users making selections within a Component system.

3.4 Paper-based Component Assembly Exercise

The exercise shown in Figure 3.8 (pages 48 & 49) is a paper-based activity that emulates the functionality of the component system prototype. Specifically, the exercise was developed to engage users in the participative design of the interior configuration for a single-level condominium floor plan.⁸ Our goal is to find the appropriate

limitations in scale, scope, and complexity of the component system in the context of participative design. This exercise explores what types of constraints are necessary within the interface to make more accessible the complex representational structure of the components. Again, components are the most atomic parts that make up a space. These would be things like lights, tables, chairs, doors, windows, sinks, etc. But to support a more simplified design process, the *exercise* allows the user to manage assemblies, rather than single components.

In the exercise, users select room layouts from a library of paper cutouts. The library has a variety of different options for each room function, and each cutout describes a unique room arrangement. Selected cutouts can then be placed onto a schematic floor plan that is divided by colors into separate zones. The zones correspond to the different types of rooms in the library: the Master Bedroom, the Master Bathroom, the Kitchen, the Living/Dining area, a Second Room, and a Second Bathroom. Each zone has a sleeve into which the cutout may be placed, to keep the composition organized.

Not all of the paper cutouts can connect to each other, because doorways and sizes do not always line up. The individual rooms were designed to combine in specific ways, yielding a more constrained solution space that allows for multiple output configurations for each of the following basic floor plan types: Loft, 1 Bedroom, 2 Bedroom, and 3 Bedroom.

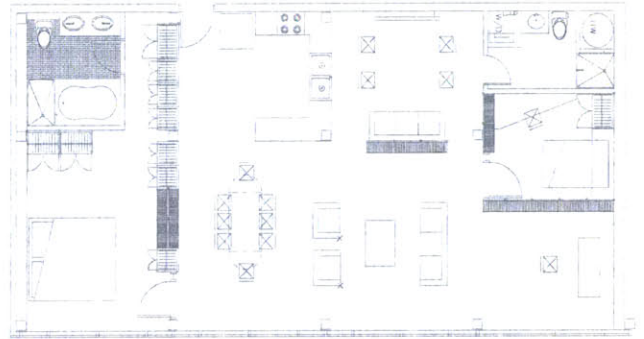
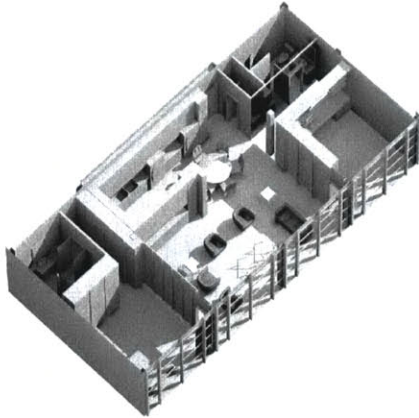
Therefore, the primary elements of the exercise are the schematic floor plan that functions as a workspace for construction, and a library of assemblies that can be placed into the floor plan to make designs (Figure 3.8). To make the exercise more approachable, the library of assemblies is organized into separate sheets, one sheet per zone, color-coded to match up intuitively with the colors of the zones themselves. Within each sheet of the library, each assembly is shown in multiple representations: a plan view, a simplified diagram in plan view, a 3D axonometric view, a brief textual description, and a photo-like rendering.⁹ The plan view is actually a cutout, and it may be detached, and placed into the schematic floor plan to create a design.

In each of the provided library sheets, eight to ten different options for each room, with each option shown in the five views

Following Page:
Figure 3.8 Component Assembly Exercise
Utilized in user studies

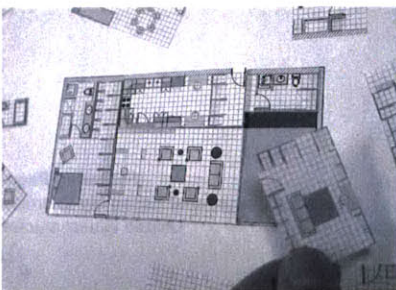
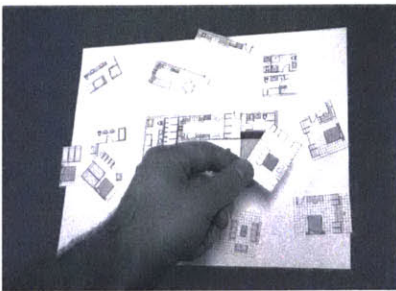
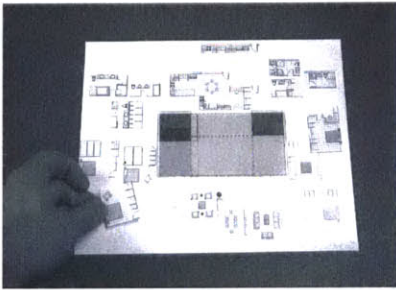
Component Based Representation

exercise developed for user study



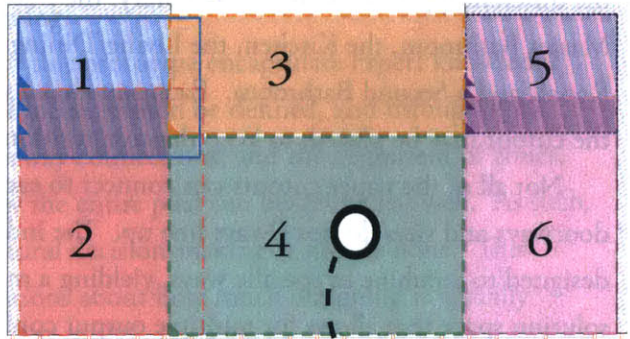
Doing the Exercise

Here's a series photos showing a user working through the placement of rooms into the schematic plan



Schematic Workspace

From the standard plan above, this schematic plan is where the users place their room selections. color code matches library cards



1. Master Bathroom
2. Master Bedroom
3. Kitchen / Entry
4. Living / Dining
5. Bathroom
6. Bedroom / Study

Detail of Library Card

This is Living Room #1
Showing the multiple representations:



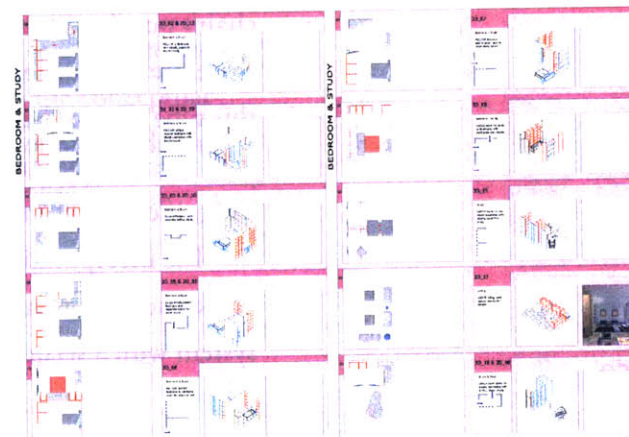
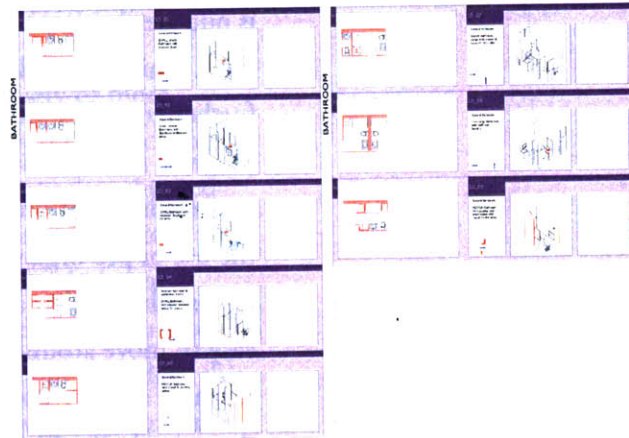
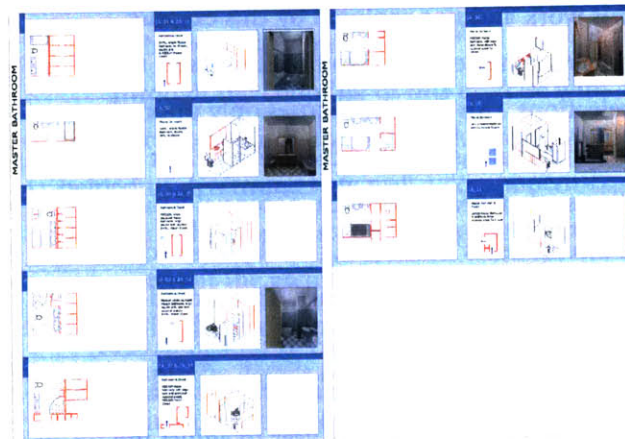
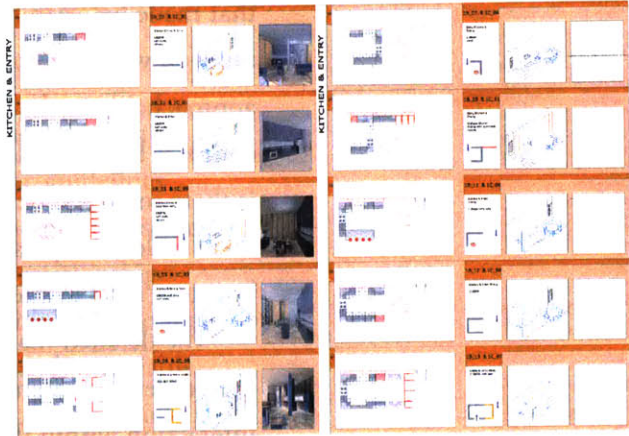
*This exercise was designed by Kent Larson, M Giles Phillips, and Carla Farina
All materials and layouts were created by Carla Farina*

Getting Started: Constrained Design Context

The exercise was limited to the interior configuration of an apartment unit (left). For each type of room, a collection of layout options was made available.

Component Library

Each type of room was given a color coded Library Card within which each of the options was shown. Each option was shown in the following five views: floor plan, diagram, text description, axonometric, and rendering



described above, yield a total of 40 or 50 distinct representations. Is that too much, or can users effectively scan through the options made available to them? How many choices do users want access to, when it comes to specific rooms? Or should users be given a choice of granularity in the final implementation?

In the exercise we have explored in particular the notion of assemblies as user-manageable groupings of components to form a basis for a practical search comparison.¹⁰ This means that the only type of connection that users can designate through our exercise is that made between rooms. The fundamental question this approach raises is: what level of granularity do people want access to or find approachable in the context of architecture? Is it easier for people to compare between plans, or to compare between rooms, or objects, or is there some less defined zone comparison that takes place between these scales of representation? In other words, is our exercise too limiting, too flexible, or just right?

The detailed manipulation by the user of the components within the predefined assemblies is taken to be a second stage of the user's interaction within the component system itself. It is a process that is outside the scope of this work and not explored by this exercise.¹¹ So, while the underlying system prototype would allow users to further configure their selections, this exercise focuses only upon the inherent searchability of the default assemblies themselves. But this separation is not, in reality, such a clear one. One potential issue with the use of pre-organized assemblies is that during the basic selection stages, objects that are included in the assembly and that would be easy to move around may be perceived by the user as inflexible. For example a room assembly may include a chair at an angle that annoys the user. In application, it would be easy for the user to adjust the chair's position after selecting the assembly, but would the chair being there prevent the selection of the assembly in the first place? Perhaps this type of issue begins to disappear as the user gains more familiarity with the system, but it should not be assumed that the distraction is insignificant within the process of selecting rooms.

The incorporation of schematic zones as a part of the exercise's workspace represents, in a sense, an overloading of the component system with a secondary schematic representation. The zones, as

an interface component, are not necessary within the underlying component system but are non-trivial in terms of the user's design process using the exercise; because the zones essentially tell the user what types of assemblies can go where. Even though the component specific limits of each assembly would automatically tease out any constructability issues, this is essentially an attempt to capture expert knowledge about what types of schemas work best, so that the users do not have to figure it out for themselves.

3.5 Linking to Conceptual Interfaces

However, the implementation of successful conceptual search functionality might eliminate the need for schematic guidelines altogether. As was demonstrated in the previous chapter, the search query itself can endeavor to describe the schematic relationship between rooms, and in searching against pre-assembled component plans, rather than schematic plans, allow people to seek out viable configurations directly. Additionally, the presence of schematic information doesn't affect the viability of the search tools or the searchability of the components.¹²

Through the coupling of this exercise with a variety of more conceptual exercises, the user evaluations detailed in the next chapter helped us determine how well a component-based exercise, engaged within the highly constrained language of room-level assemblies, can represent the architectural needs and values of the designer. Certainly, the decisions made are reduced in number and influenced by the specific details of the assemblies themselves. But how does this constraint alter the processes of discovery and of designing, for better or worse? Does the Component system allow people to discover their preferences; to figure out what they're looking for? Or are the conceptual query interfaces better for that? And do the conceptual exercises really facilitate any self-reflective learning, or are needs and values lost still in the complexity of the representations?¹³

1. In the context of connections between things, the scope of this research is limited to the connections inherent to an apartment interior. In terms of future work, the final chapter will speculate about how the connection definitions herein may or may not scale into considerations of site, neighboring structures, and solar orientation.
2. In many contexts, OOP is more of a methodology choice, and not necessarily constrained by the language itself. While purely object-oriented languages like Java and .NET have emerged and become quite popular, other programming languages exist that allow for both procedural and object-oriented programming techniques, and are popular as well.
3. The structure of OOP maps well to human thought, primarily by taking advantage of the same underlying object / physical metaphors. Speaking to human thought, Michael Reddy asserts the inherently low-level physical metaphors of natural language form a basis for the structure of ideas (Reddy), and Winston speculates that people's actual knowledge structure involves relatable, autonomous parts. (Winston) So it seems to make sense that virtual structures, organized together into an assembly of objects, would be easier for people to think about.
4. XML, being a structured document format that allows items to be both identified with properties and related, has proven to be both popular and useful, though it's a simple hierarchical structure and doesn't describe data types inherently.
5. If we believe that any representation is actually an index, then the elements which each representation describes are deemed important by inclusion alone.
6. The component system prototype was designed by Kent Larson, M. Giles Phillips, and Carla Farina. The system itself, as detailed in chapter 6, was developed by M. Giles Phillips.
7. Kent Larson, Carla Farina, M Giles Phillips, MIT House_n
8. The evaluated apartment floor plan context also constrains the problem to a single level. While the staircase connection between levels would naturally be drawn between points of entry to each level, this work makes no assumption that the addition of multi-level functionality would be straightforward. However, our inquiry is aimed at the most fundamental aspects of composing plans and as such, one level is complicated enough.
9. The assemblies are intended to be configurable, in that the components within them could be moved around as long as connections and clearances weren't violated. But alas, the specifics of the interface to support this next step of designing are outside the scope of this work.
10. As will be touched upon when the more conceptual interfaces are described, the schematic representation here really represents an alternate conceptual search, albeit one that relates closely to a diagrammatic

conceptualization tool that is evaluated. The direct searchability of the conceptual representation is analyzed in Chapter 6.

11. The assemblies are intended to simplify the design process, basically by having some of the architectural decisions pre-made. However, not all assemblies are created equal: rooms like bathrooms and kitchens tend to be both more complex in terms of the immovable fixtures and also less configurable. We did preliminary evaluations to help organize the various parts of the component exercise and also helped us to identify where the assemblies were too granular. We had for example initially separated the kitchen into two different zones, each of which needed to be selected individually, and in so doing managed to confuse even ourselves. This was revised to be one zone in the user studies.
12. Each of the representations is provided to begin to sort out, through user evaluations, what type of information is useful for the various decisions made in designing. More complicated rooms may for example benefit from associated diagrammatic views. Another dimension of this inquiry is the potential problems with some of the representations, like the colors and detail of renderings adversely affecting decisions or confusing the user.
13. Donald Schon describes self-reflection as a critical part of the learning process when trying to solve a problem that is new or has unexpected dimensions or complications.

4 The Query: Conceptual, Constructive Interfaces

4.1 Why utilize Conceptual Interfaces?

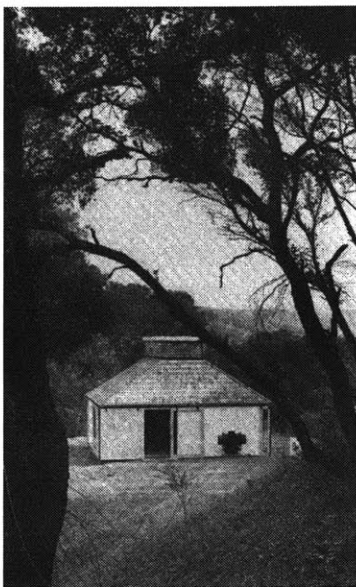


Figure 4.1 Moore House, Orinda by Charles Moore, 1961

Addressing homeowners, Charles Moore notes: “our task, now, is to clarify choices, to focus your energy so that it will not all be spent trying to find a way through the muddle of building decisions, but instead can be used to bring your own personal concerns to bear.” (Moore, viii) That motivation is what this chapter is all about, because the numerous and complex options that a homeowner must search through certainly create a muddle of building decisions. Are there clearer, more conceptual interfaces that would allow people to better describe their concerns, and which could also be implemented as a query interface for our design search? This chapter describes, in detail, four different query interfaces which utilize unique conceptual structures to frame design problems. The output generated by each of these interfaces would be a search query, which can then seek out either component system configurations or pre-indexed “flat” representations like floor plan images.

The queries are intended to be diverse and to provide more than one type of representation to search with; further, the queries are intended not only to find valid design solutions, but to help users learn about and frame their own architectural needs and values. Our preliminary research showed that several problematic situations emerged when users were asked to begin designing a space without first thinking about design requirements.¹ First, there was difficulty with problem separation. In situations where the user could identify a specific need, they were often not able to find a floor plan or assembly that adequately meets their needs, because the selection process is clouded by other non-related issues with parts of the configuration, like the presence of additional, unwanted components, relative sizes, or organizational variances. And more fundamentally, people had difficulty *discovering* specific needs when only looking at the component representations. The primary reason for this seems to be that the more literal representational structure does not necessarily encourage associative thinking that relates ideas from other

conceptual domains. So, the design process becomes one of option selection; there is greatly reduced potential for the expression of concerns, or the emergent concerns that were not yet thought about. Many people benefit, for example, from thinking about narrative sequences of actions to reflect upon how they might actually exist within various spaces.² This allows them to think about the space in different ways, and to remember and organize their needs. But when working with the complex and pre-designed assemblies, the user becomes absorbed in the literal dimensions of the pre-arranged configurations, just as Moore describes, making it hard for them to focus on what is actually important.

Another issue in working with a modeling environment, or even component-based representations like BIM, is that specific trade-offs are created between spaces in a highly contextualized way. For example, a larger bathroom might mean that the kitchen will have to be smaller. This then becomes a trade-off that the user has to think about, basically to decide which room is more important in the specific context considered. These targeted tradeoffs instigate an iterative process that helps make the process more interesting, but they represent context-specific and non-generalized decisions that might not make immediate sense to or be important to the user, particularly in terms of their own needs and values. And even more problematic, in terms of a search algorithm, is the fact that the context-specific trade-offs cannot be generalized to other potentially searchable contexts, or other content providers: the trade-offs simply would not be the same. Furthermore, the more literal representations force users to think about space according to already designed and probably standardized configurations, thus potentially limiting the ability of the system to encourage the discovery of new, emergent configurations. All of these limitations suggest that an interface to enable search by designing must allow for the more fundamental; indeed, the more conceptual exploration of space.

4.2 Four Conceptual Interfaces

This is where the search process becomes useful, in allowing people to frame problems and seek out solutions simultaneously, free from the distractions in the structure of the more literal representations they are seeking out. Therefore, as queries, the conceptual tools described here are intended to do two critical things: encourage associative thinking to help frame spatial problems, and to create output that is able to be generalized and therefore useful for searches. While the viability of each conceptual interface is evaluated separately, they have been designed to represent conceptual information in terms of a common underlying structure.³ Each conceptual interface was designed to be viable for certain types of problem framing, so that different types of representations could be selected by users based upon personal preference, or users could specify search criteria within multiple representations to generate a more complete, layered query.

Four distinct querying interfaces have been designed and evaluated for this study, to offer users with more than one option for expressing themselves. The interfaces are detailed in the following sections.

4.2.1 Text-Based Checklists

The text-based interface is provided to evaluate the effectiveness of natural language descriptions in helping conceive of spatial arrangements. This interface was evaluated as an example of a standard, contemporary “option selection” search interface, against which the other interfaces could then be compared.

This interface tests the viability of a natural language description as a way to frame spatial concepts. In this example, each functional room/zone has a series of short, textual descriptions that are selectable as search criteria; these options would correspond roughly to the assemblies that exist within the system. Of all of the query interfaces, this one is probably the most like a traditional search engine, particularly in the context of home searches, but it is also

problematic.⁴ First, natural language representations have well-documented semantic issues; suffice it to say that the context dependency of spatially descriptive terms like “large”, “near”, or “open” make them confusing to the user, particularly without some sort of visual device or scale to quantify things. Also problematic is the description of attributes that are more qualitative in nature: identifying, for example, a stylistic preference using words alone, can be a challenge. In developing an algorithm for mapping natural language to relational database structures for IBM, John F. Sowa notes the semantic ambiguity in a text string and proposes that intermediary representations may serve as a “semantic basis” for extending written language, as a way to both manage and explain inference. (Sowa 336) But this solution requires some sort of natural language processor to translate the natural language into a graph-like representation, (Sowa, 347) and might still require humans to be involved in programming / data selection to actually work (Sowa 356)

And so it goes, that apart from the inherent semantic issues, the text-based conceptual representation does not scale that well either. For example, in the case of Component systems or BIM as a target resource, available assemblies would have to be annotated as they were defined, likely manually, with correctly summarized descriptions of their functionality. Being highly semantic, it is not plausible that a system, given today’s technology, would be able to automatically generate something like this from component tags.⁵ Each new description would need to be pushed to the query interface somehow, to present the option. Two scalability problems thus exist. The representation itself is not scalable, because potentially limitless verbal descriptions must exist in the system for each space type to be searchable: the user might have literally thousands of options. And the descriptions themselves must be manually designated for each assembly. These descriptions would be disconnected from the structure of the representation itself, creating an exponential problem of manageability. The descriptions are an ad-hoc attempt to find representations that make sense to both people and machines, but it’s an obvious point that inherently, natural language descriptions make more sense for people than they do for machines.

That said, text-based queries map most easily into the most common conception of the way a search is structured, that of: enter text, hit button, select result. This type of representation might also become much more scalable, and thus viable, as technology and language processing improves. And most people may simply be more comfortable saying things in words than using any sort of visual or abstract interface, regardless of how hard those words are to compute with. If this is the case, then we would certainly be glad to know it, and better served focusing our energy in areas other than a study like this one, which evaluates the assertion that people can, and want to do more than speak in words. For all of these reasons text-based queries are important to evaluate, even for comparison purposes alone, and are thus included.

4.2.2 Role-Playing Metaphor

The role-playing metaphor is a listing task that encourages users to conceive of space in terms of their possessions by imagining they are packing for a move. This task provides users with a listing step and then a reflection step. The motivation for separating these steps is to encourage reflection by having the person first describe certain facts and then reflect upon that which was described.

This interface uses items, or more specifically, things that go in a certain room, as a way to frame spatial concepts. The activity utilizes a high-level metaphor as a way to engage the user into a situational role: that of packing a box. The packing metaphor basically gives the user a way to start thinking about what functional needs they have of a place, by thinking about all of the stuff they would have to put there. Items, regardless of size, shape, weight, and fragility, go in the box. Room characteristics, or attributes, are associated with labels that are drawn on the sides of the box. The interface thus organizes a two-part process of listing and labeling around a simple metaphorical concept. The labels given to designate room characteristics represent the most important searchable criteria.

There is a technical challenge with this interface: making items available to the user for the items list, simply because it is hard to know ahead of time what specific types of items a user has in her

rooms. The room attributes are a more manageable set and are predefined, so at a minimum these specifications would be searchable. But the item list would be useful to have as well, even if it was only partially searchable.⁶ Within the prototyped interface, custom labeling could be added by the user, with the caveat that the custom labels might be meaningless in terms of the search comparison. But, given that the user had thought about the item, and likely derived a searchable room attribute from the conception of that item, it is quite possible that custom items would be represented, at least in part, by this query structure, via the associated room attributes for which they illuminated the need.

The attributes are therefore not customizable, and would have specific quantified criteria that would be meaningful to the search operation. To be clear, each attribute option should include a description of the quantified criteria that the system would infer from that attribute. As will be seen, volunteer subjects referred to these detailed descriptions relatively frequently during the exercises, sometimes revising their selection of the attribute based upon those descriptions. This is an important point to make because, at the highest level, an attribute like “spacious” has the same semantic issues as those described in the Checklist interface. But through the associated and human-readable description, the meaning is clarified: these descriptions serve as a semantic definition, and make the term explicit. So, while it is the attribute word that attracts the user to make the initial selection, the semantic definition tells the user what the word means to the computer, ahead of time. These semantic definitions are plausible because in the context of the study they are all spatial in nature, and there are a relatively small number of words. And as will be shown, the user evaluations highlight where the semantic definitions are either confusing or inadequate.

4.2.3 Activity Sequencer

The activity-based interface is similar to the metaphorical interface but instead focuses upon a sequencing task that allows users to explore how they flow through a space and identify architectural values through that exploration. The motivation for this interface

was to help homeowners identify needs by organizing concepts around tangible flows through space. To help users discover needs, this interface allows the user to first describe and subsequently reflect upon a sequence of activities.

People are very good at thinking about things in terms of sequences; many psychologists suggest that low-level mental structures directly correspond to narrative or sequential structures.⁷ The Activity Sequencer explores the particular effectiveness of activities as a way to frame spatial concepts. In *The Place of Houses*, Charles Moore presents to the potential homeowner a list of options designed to encourage their conception of people or things as they flow through a home.⁸ These flows relate to activities, groceries, papers, trash, water, electricity, and so on. What's interesting about this approach is that the conception of space is organized into a narrative sequence of steps; this organizes serially the various interactions that happen within the home. This interface was developed with a very similar idea in mind: to organize home preferences around a sequenced activity.

Users first list out each of the activities, then locate those activities within specific rooms in their current living space. Once the activities are located, users can identify the transitions they make from activity to activity. Then, users reference this complete activity description to identify important room qualities, in terms of that activity. Once those qualities are identified, the user may go back and revise the activity sequence to better meet their needs.

Like the packing interface, this interface really involves two phases: first listing, which is followed by reflection. While the separation of the task into discrete steps as described in the step-by-step summary makes the activity sound a bit overwhelming, it actually helps to make the process manageable because each step is very clear and specific. The intention in both this and the metaphor interface is to give the user a chance to start conceiving of space and functional needs in terms of things or routines that are well-known to them.

4.2.4 Floor Plan Diagram

The diagramming task allows users to conceive of space in terms of an abstract spatial description. It was inspired in part by Kevin Lynch's work which suggests that people understand space in terms of basic elements that can be represented diagrammatically. A further motivation was to provide and evaluate a more constructive interface where the user has more control over the "shape" of the design artifact.

This interface relies upon the effectiveness of an abstract floor plan representation as a way to frame spatial concepts. An important precedent for this particular exercise is the well-known research of Kevin Lynch, which illuminates a common descriptive language for people's conception of space as they interact with(in) it. Lynch identified five terms for this descriptive language: districts, edges, nodes, landmarks, and paths. (Lynch) This interface presents the user with a diagramming activity, with specific types of elements that can be placed in or between the cells of a grid. The grid cells correlate with Lynch's notion of districts, which are, in this case, rooms that the user can associate with one or more functions. In the space between the rooms, which is closed by default, specific edge conditions or connections can be specified: the user can place either doorways or complete openings. These specific connections allow for the emergence of paths and nodes, both fundamental spatial concepts identified by Lynch.

Because of the grid, the diagram is highly constrained: the zones that are used for rooms are rigidly defined as cells. Pilot studies done with more flexible diagrammatic representations indicated that users tended to get flustered or confused if they weren't given a bit more constraint.⁹ So the constraints in this interface have two specific functions: to make the generated query be something that is plausible to search with and to make the diagramming something that is manageable for everyday users. In other words, to make the representation something that is meaningful to computers as well as users, rather than one or the other.

The interface gives the user the option to first diagram their current living situation, and then to make a few improvements upon that diagram as a reflection of their preferences. Therefore, it roughly follows the precedent established by the Packing and Activity

Sequencer interfaces, of first describing something, then reflecting upon what was described to arrive at needs and values. In this particular case, the purpose of the descriptive step is to give the user a chance to get used to the diagramming language by thinking about a space that is well known to them. However, for the user, describing his or her current living situation might not be a good foundation for describing what he or she really wants. While the revisions made in the reflection stage could be very extensive, users may constrain themselves to the descriptive diagram, being unclear on how they might revise it without breaking something. The interface should therefore allow the users to start from scratch, if they prefer to.

Additionally, this interface is unique when compared to the others in terms of the flexibility of its descriptive workspace, and for its ability to let users more directly construct things. Granted, the representation is highly constrained by the fact that they are working within a grid and can only place a few items, but even so, the constructed query representation is much more of a sketch than the other activities. This means the query itself becomes something like a workspace, where various configurations may be thrown together over successive iterations. This is something that is very much at the heart of designing. The risk, which will be at the heart of this evaluation, is that aspects of this workspace query might then go unrepresented in the search algorithm, and therefore cause the search to fall short of the user's expectation of what she will get. In terms of a search tool in particular, this means there is a significant risk that the application would not be effective

4.3 Conceptual Interface Considerations

There are a few general considerations related to these conceptual interfaces apart from those touched upon in the detailed descriptions. The first of these is one of expectation, in terms of what the user thinks is important about their descriptions. Are the aspects of the composition that are meaningful to them the same things that are meaningful to the search? And does any sort of prioritization option need to be provided to let the user highlight specific parts of their query as more important?

Secondly, how approachable are the problems, both in terms of the clarity of the interface and the nature of what the interfaces ask people to think about? While efforts have been made to keep each step simple and straightforward, overall clarity is a core concern of the study. An analysis of the interface's approachability must be able to tease out whether it is the fundamental nature of the task, or if it is the specific details of the task as presented that makes emergent any unforeseen issues.

4.4 Searching with Conceptual Queries

An important consideration to touch upon here (detailed in Chapter 6) is how the queries generated by each of these conceptual interfaces might be mapped to searchable formats. For example, the items that were listed in any of the conceptual exercises could be mapped directly to matching objects in an object-oriented component system. In this case, the matching would be based upon a categorical "type" structure that defines what the specific objects are used for. The search would also need to consider the hierarchy of what the object relates to, where possible. For example, being *within* a particular room is important to consider as part of the matching criteria. Transitions between spaces identified either by the diagramming task or the Activity Sequencer would need to be mapped to specific connections between rooms in the component system, and possibly even specific object configurations within those assemblies. As such, the standardized output must, at the very least, describe container objects like rooms, autonomous objects like furniture that are organized within those rooms, and the connections between the rooms themselves.

For activities, mapping isn't quite as straightforward, but might also match to specific items where possible or necessary. In some cases it will be an obvious specification, not necessarily needing representation, as in the case of the morning routine: it should be fairly obvious that "showering", tagged by the user as something that happened in the bathroom, will also happen, more specifically, in the shower. But all of the bathroom assemblies and therefore any potential match in the component system are going to have a shower

in it – the search doesn't have to be so explicit. Further, simply by the fact that the object goes unrepresented in the room specification, it is likely that the user will reflect upon the fixture used in the activity and further specify its qualities if that specification is important. So if there is something important about the shower, the user does get the chance to say it.

4.5 Exercises Developed to Evaluate the Interfaces

This section details the design of paper-based exercises that were prototyped to evaluate each of the four distinct querying interfaces described above. One exercise was created for each interface. All of these exercises are paper-based prototypes where paper cutouts and tags can be arranged on a special worksheet. For each exercise, there are paper tags (shown in Figure 4.2) that represent the following things: activities done in a space, physical (bodily) transitions made in space, objects, room names, and room attributes. These tags are meant to correspond to drop-down options in an application interface. The different exercises allow the users to organize these various tags in different ways, on different worksheets, and in different scopes. The first exercise presented, to evaluate the Text-Based Checklist interface, is simply a checklist; therefore no tagging takes place. And the final exercise, which evaluates the Floor Plan Diagram interface, introduces additional diagramming cutouts, which will be detailed in that section.

In terms of the exercises, there is an issue related to how much the user's compositions are influenced by any examples that might be given to them during the evaluations; given in an effort to clarify the exercises. For example, in the diagramming exercise, the users are shown an example diagram to help them see how all of the elements can be placed. (Appendix 1, A1.4) In some of our initial studies done during the prototyping of the exercises, the diagram that the user composed immediately afterwards had significant similarities. Were they working towards a mental image of the example diagram without knowing it?

Conceptual Exercise

Tag Cut-Outs for Exercises 2-4

Activities

Bathing	Cleaning	Cooking	Computer Use
Dressing	Drinking	Eating	Evacuating
Exercising	Groceries	Grooming	Hobby/Craft
Listen to Music	Partying	Reading	Relaxing
Resting / Nap	Shaving	Showering	Sitting
Sleeping	Studying	Video Games	Waking Up
Watching Movies	Watching TV	Working	Brush Teeth
Snoozing	Coffee	Check Weather	Keep Sleeping

Transitions for Activity Sequences

Move to a Different Part of Room	Move Through Opening	Change in Orientation	Move Through Multiple Spaces
Move Through Doorway	No Change in Position / Orientation	Change in Position	Change in Position

Room Attributes with Definitions

Accessible Entrance/egress requires no open floor space, steps or thresholds	Country Style tagged by component	Private Access controlled and locked, no view or egress	Spacious 75% of floor area is clear unobstructed
Closed Layout Entrance/egress controlled or obstructed with other	Regional Style tagged by component	Public Opening to adjacent spaces without door	Traditional Style tagged by component
Efficient Entrance/egress controlled or obstructed with other	Some Light Entrance/egress controlled or close to wall side	Open Layout Entrance/egress requires no open floor space, steps or thresholds	Contemporary Style tagged by component
Small But Comfortable Clear height, circulation space 75% or more of floor or area needed	Abundant Artificial Light Presence of multiple lighting components	Abundant Natural Light Presence of windows, patios or sky area with view	Lots of Counterspace

Notes

The room attribute tags included semantic definitions to clarify their meanings. The room tags themselves simply designate functionality and can be grouped together to create multifunctional spaces. Users were permitted to write in additional items if necessary, using the blank Item tags.

Items

Bathtub	Bathroom Sink	Shower	Bookcase
Jacuzzi	Toilet	Bed	Home Theater
Office Desk	Sofa	Closet	Books
Bar	Kitchen Sink	Oven & Stove	Cookbooks
Love Seat	Coffee Table	Dining Table	Electronic Devices
Shelving	Kitchen Pantry	Shelving	Movies
Flat Screen TV	Small Coffee Table	Small Dining Table	Dish Rack
Island	Pots and Pans	Linens	Telephone
Coffee Machine	Frozen Food	Dry Food	Chair
Digital Media	Clothing	Craft Supplies	Knick Knacks
Misc. Equipment	Pictures		

Rooms

Bathroom	Bedroom	Closets	Dining
Kitchen	Entry	Family	Half-Bath
Hallway	Living	Media	Study

Figure 4.2 The Tags Used for the Exercises

4.5.1 Step by Step Exercise Summary: Text-Based Checklists

The checklists are shown in Figure 4.3 (Page 68). Users were asked to review a text-based checklist and for each type of room, select the one option that best worked for them.

4.5.2 Step by Step Exercise Summary: Role-Playing Metaphors

The exercise board is shown in Figure 4.4 (Page 69). Users were asked to imagine that they'd just bought a new home and were packing up their things. For one important room, they were told to first identify the room and then list all of their possessions for that room using paper cutouts that had item names on them. For the purposes of this paper-based exercise, blank tags were also provided so that the users could write any item-label they wanted. Finally, they were asked to reflect upon the list of items and identify important room qualities using the red attribute tags shown in Figure 4.2.

4.5.3 Step by Step Exercise Summary: Activity Sequencer

The exercise board is shown in Figure 4.5 (Page 70). This exercise has 5 distinct steps, each of which involves the placing of printed labels shown in Figure 4.2 into the workspace shown. For each step, a pre-selected list of keyword cutouts was made available to the user, who can then insert them into the correct sleeve. The first step is to place each of the activities of their current morning routine into the top of the circles. The activity tags allow users to organize things like "waking up", "showering", "drinking coffee", and so on. The second step is to go back and tag, in the bottom part of the circles, the room each activity happens within. Next, the user identifies the bodily transition between steps, in the green bars between the circles. After the transitions have been labeled, the fourth step of this exercise asks the user to begin reflecting on the activities that are sequenced and identify room attributes that are important or desired, in light of these activities. The fifth and final step allows the user to go back and revise either transitions or rooms, if they think they could improve the sequence by doing so.

As a control, only one activity was made available to the users who came in as volunteers, although in application this interface is meant to be more general. The selected activity was “your morning routine”: users were asked to list out each step of their morning, in chronological order. The morning routine itself was chosen because it is something that everybody has to do and it is something with fairly common activities but also a lot of individuality. Additionally, the morning routine tends to have at least two or three rooms in its scope, and therefore the user is far more likely to think about different rooms, and as we will see, the transitions between them.

4.5.4 Step by Step Exercise Summary: Floor Plan Diagram

The exercise is shown in Figure 4.6 (Page 71). Users were shown a set of diagramming pieces which they could use to make an abstract floor plan within the cells of a grid that was drawn on the workspace. Users were told that they could place openings or doorways between cells. In addition, a “water wall” element, which represents a wall that has plumbing connections, was provided, to be placed alongside one edge of every kitchen or bathroom cell. The user was encouraged to use as few water walls as possible, in an effort to inject a bit of guidance into the diagramming task: minimizing the number of water walls ought to make the layout more efficient. A main entry could also be placed.

Users were told to first describe their current situation, and then to make one to three improvements to the design. If they lived in a space with more than one level, they were simply told to pick a level. Users were told that they could start their diagram from scratch if they felt like that would be easier. And as will be seen in the summary of user evaluations in the next chapter, this task turned out to be surprisingly approachable.

Conceptual Exercise 1

Text-Based Checklists

1. Master Bathroom

- A. Small, Simple configuration
- B. Medium w/ Laundry and Larger Sink
- C. Medium w/ Water Jet Bath
- D. Medium w/ Water Jet Bath and Extra Closet
- E. Large w/ Sauna and Steam Room
- F. Large w/ Extra Closet

2. Master Closet

- A. Small, enough storage for one person or two people who don't have too many clothes
- B. Medium, adequate storage for two people
- C. Large, spacious closets with quite a bit of shelving, good for two people who have lots of clothing

3. Master Bedroom

- A. Smaller, 12 x 14, with a simple configuration
- B. Medium, around 14x14
- C. Medium w/ some built-in, additional Closet space
- D. Medium w/ space for flat screen TV and equipment
- E. Larger bedroom, around 16 x 14, w/ Additional Seating Area and open floor space
- F. Large w/ space for flat screen TV and equipment
- G. Large, Loft Style: seating area, with open connection to living space, and open floors

4. Second Bathroom

- A. Smaller, basic bathroom w/ Shower
- B. Medium bathroom with Shower and also a Washer and Dryer unit just outside
- C. Larger bathroom with either separate water closet, additional storage, or a Washer and Dryer just outside

5. Kitchen + Entry

- A. Open, Loft Style: linear and open to living space
- B. Open, Loft Style w/ Dining Table added
- C. Open, Loft Style w/ Dining Table and Separate Entry area
- D. Open, Loft Style w/ Bar between kitchen and living
- E. Galley Style w/ Separate Study area
- F. L-Shaped, Small w/ Separate Dining Table area and Bar
- G. U-Shaped, Small w/ Separate Dining and Entry Closet Space
- H. L-Shaped, Large w/ Bar and Dining Table
- I. Closed (enter through door), Large, with Study space

6. Living / Dining

- A. Large Living Space, Open to all Adjacent Spaces
- B. Large Living Space, With Shelving Along Edges so it's more closed off
- C. Medium-Large Living Space w/ Separated Office area
- D. Medium Living Space and Medium Dining Space, Open to each other
- E. Smaller Living Space with Two Additional Bedrooms (3br total)

7. Study / Second Bedroom

- A. Small Bedroom w/ Separate Medium-Sized Study Area
- B. Medium Bedroom w/ Smaller Separated Study
- C. Large, single Room; multi-functional Bedroom/Study
- D. Large Bedroom that can also function as a study
- E. Two Medium Sized Bedrooms w/ Study Desks (req. 5E)

Instructions

For each of the rooms (1-7) review the options given and check the one that best meets your needs.

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date _____
Session _____
Task # _____
Sheet # _____

1. Master Bathroom

- A. Small, Simple configuration
- B. Medium w/ Laundry and Larger Sink
- C. Medium w/ Water Jet Bath
- D. Medium w/ Water Jet Bath and Extra Closet
- E. Large w/ Sauna and Steam Room
- F. Large w/ Extra Closet

2. Master Closet

- A. Small, enough storage for one person or two people who don't have too many clothes
- B. Medium, adequate storage for two people
- C. Large, spacious closets with quite a bit of shelving, good for two people who have lots of clothing

3. Master Bedroom

- A. Smaller, 12 x 14, with a simple configuration
- B. Medium, around 14x14
- C. Medium w/ some built-in, additional Closet space
- D. Medium w/ space for flat screen TV and equipment
- E. Larger bedroom, around 16 x 14, w/ Additional Seating Area and open floor space
- F. Large w/ space for flat screen TV and equipment
- G. Large, Loft Style: seating area, with open connection to living space, and open floors

4. Second Bathroom

- A. Smaller, basic bathroom w/ Shower
- B. Medium bathroom with Shower and also a Washer and Dryer unit just outside
- C. Larger bathroom with either separate water closet, additional storage, or a Washer and Dryer just outside

5. Kitchen + Entry

- A. Open, Loft Style: linear and open to living space
- B. Open, Loft Style w/ Dining Table added
- C. Open, Loft Style w/ Dining Table and Separate Entry area
- D. Open, Loft Style w/ Bar between kitchen and living
- E. Galley Style w/ Separate Study area
- F. L-Shaped, Small w/ Separate Dining Table area and Bar
- G. U-Shaped, Small w/ Separate Dining and Entry Closet Space
- H. L-Shaped, Large w/ Bar and Dining Table
- I. Closed (enter through door), Large, with Study space

6. Living / Dining

- A. Large Living Space, Open to all Adjacent Spaces
- B. Large Living Space, With Shelving Along Edges so it's more closed off
- C. Medium-Large Living Space w/ Separated Office area
- D. Medium Living Space and Medium Dining Space, Open to each other
- E. Smaller Living Space with Two Additional Bedrooms (3br total)

7. Study / Second Bedroom

- A. Small Bedroom w/ Separate Medium-Sized Study Area
- B. Medium Bedroom w/ Smaller Separated Study
- C. Large, single Room; multi-functional Bedroom/Study
- D. Large Bedroom that can also function as a study
- E. Two Medium Sized Bedrooms w/ Study Desks (req. 5E)


Exercise: Written Description

Figure 4.3 The Checklist Specification Exercise

Conceptual Exercise 2

Role Playing Metaphors


Room Name: _____



Room Attributes

Items in Room

Room Name: _____



Room Attributes

Items in Room


Instructions

Imagine that you're packing up all of the important items in one of your rooms. List out each of the items you'd need to pack. Reflect upon the list, then label the box with the characteristics the room would need to have to meet your needs

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date: _____
Session: _____
Task #: _____
Sheet #: _____

Room Name: **Bedroom**




Room Attributes

- Abundant Natural Light
- Contemporary
- Spacious
- Open Layout
- Efficient

Items in Room

- Bed
- Office Desk
- Chair
- Electronic Devices
- Closet
- Books
- Clothing
- Misc. Equipment

Room Name: **Living**



Room Attributes

- Contemporary
- Seamless
- Clean Layout
- Efficient

Items in Room

- Living Chair
- Chair
- Home Theater
- Bookcase
- Sofa
- Small Coffee Table
- Stove
- Books

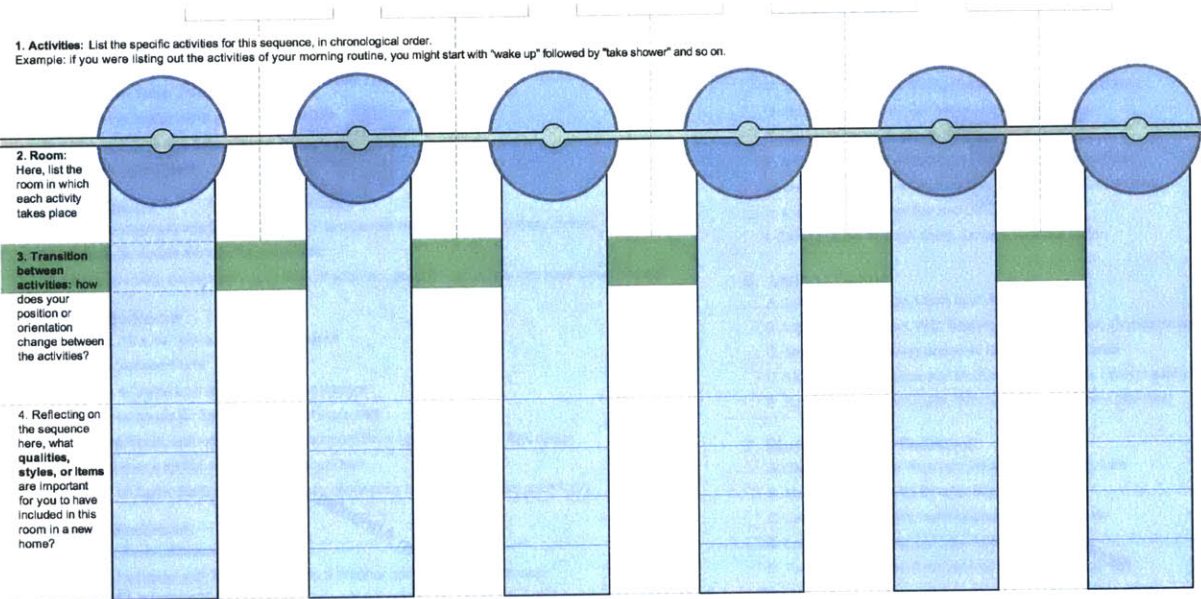
Example, created by volunteers during the interview sessions

Exercise: Pack up your stuff, it's time to move in!

Figure 4.4 Role-Playing Metaphor: The Packing Exercise

Conceptual Exercise 3

Activity Sequencer



Instructions

Identify and place in chronological order the activities that comprise your morning routine. Next, label the room that each of those activities takes place within. Next, identify the transition you make between each of the activities. Now that the activities are sequenced, reflect upon the properties of each of the rooms included and identify properties that are important to you. And finally, could any of the transitions be changed to improve the sequence?

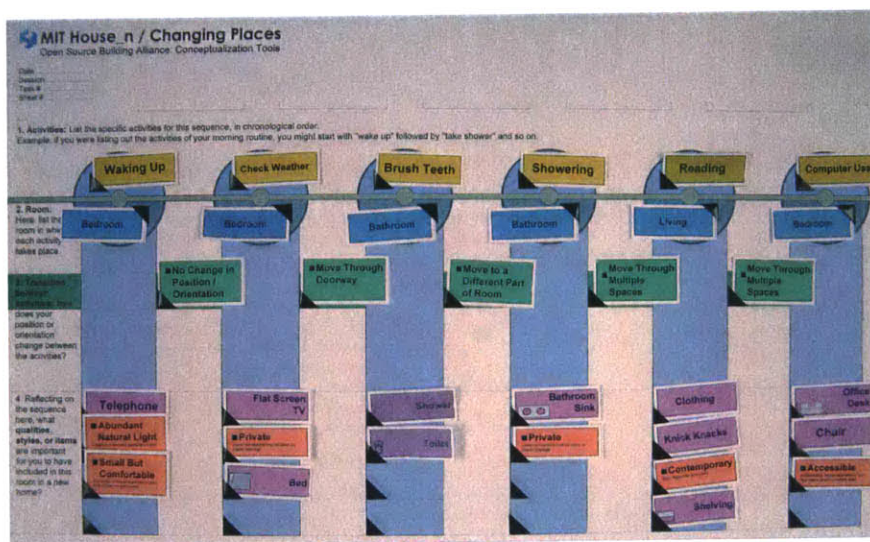


Figure 4.5 The Activity Sequencer

Conceptual Exercise 4

Floor Plan Diagram

Grid Board:

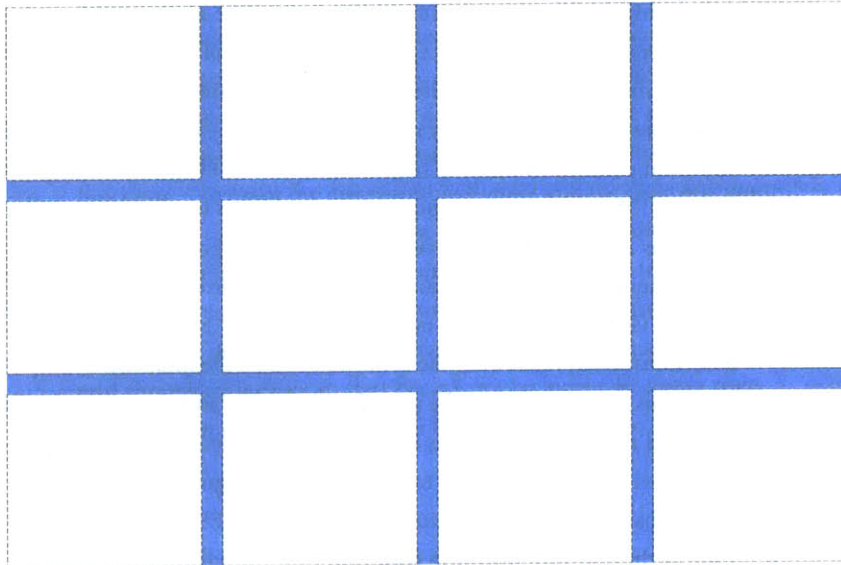
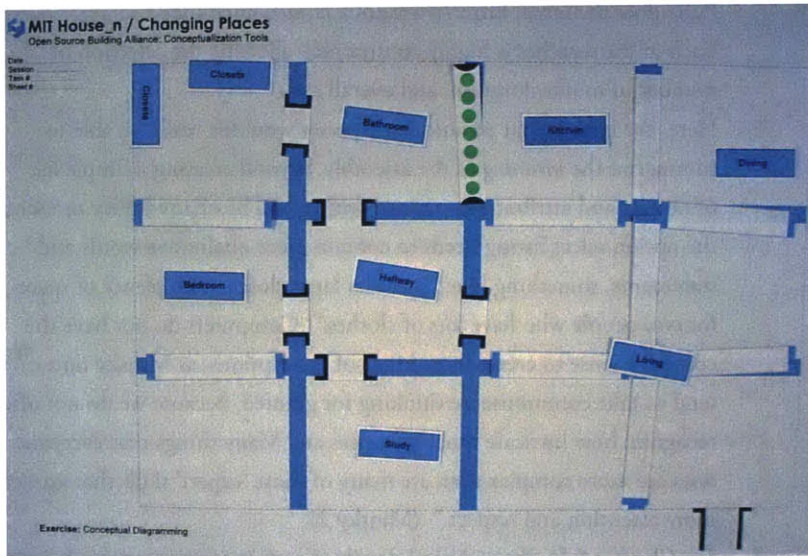


Diagram Pieces:



Instructions

Using the provided grid, create a diagram that represents your current living space, and then make 1-3 changes that would improve the space. For an idea of how the various cut-outs should be placed, refer to the example diagram. When placing water walls, keep in mind that they are expensive, and should be used efficiently.



Above: the example diagram that was provided to the volunteers.
See Appendix A1.4 for larger view

One of the volunteer's diagrams

Figure 4.6 The Floor Plan Diagram Exercise

1. Our preliminary studies involved the arrangement of pre-designed room cut-outs on a paper worksheet. The studies were done with 6 subjects; all of them were House_n affiliates. 4 of the 6 had educational or professional experience in architectural design. In reflecting on the exercise, each of the subjects noted that while some of the decisions they made were because of some preference, they made many decisions that were highly specific to the organizational logic in the options given, and not directly related to any conceptual need or value of their own. There was also no evidence that the literal room representations in the exercise allowed anyone to think of any new needs or values – only those who had identified preferences ahead of time were able to fulfill them in translation.
2. Perkins (1981) suggests that unconscious thinking is fundamental to all conscious activities, including creative ones like designing, and that this unconscious thinking is structured into sequential processes. Along these lines, C. G. Jung describes his own design process: “I built the house in sections, always following the concrete needs of the moment... Only afterwards did I see how all the parts fitted together and that a meaningful form had resulted.” As Charles Moore observes, “[Jung] had been compelled by his unconscious to build it as he did, guided by impulses deeper than those of the conscious will.” (Moore, 129)
3. Each of the querying interfaces described here will adhere to a common XML structure and output data in an XML stream. The details of this generation will be discussed in Chapter 6.
4. For example, as of 2007 the Toll Brothers site hosts a flat, simple, option selection-based search called “Design Your Own Home”: <http://www.designyourownhome.com>
Lennar does is well: <http://www.lennar.com/findhome/search.aspx>
Pulte does the same: <http://www.pulte.com/homefinder>
Each of the searches is highly constrained, allowing the selection of number of rooms, location, and overall size.
5. Here, the issue is that an automatic parser wouldn't really be able to summarize the *meaning* of the assembly, beyond creating a simple list of objects and attributes contained within. To be of any utility to users, the option select listing needs to contain more qualitative words and statements, something like: “includes large closets with plenty of space for two people who have lots of clothes.” Computers do not have the common sense to create these kinds of descriptions, as Minsky notes: “We tend to take commonsense thinking for granted, because we do not often recognize how intricate those processes are. Many things that everyone does are more complex than are many of those ‘expert’ skills that attract more attention and respect.” (Minsky 2)
6. See Chapter 6 for the technical details of each implementation. It covers the custom tags within each of the search queries.
7. See Minsky's assertions in *Society Of Mind*, specifically Chapter 25

5 *User Evaluations and Findings*

The evaluations described herein were used to test out the viability of the component-based and conceptual exercises as tools for everyday people. The viability of each exercise was measured in terms of its ease of use, how enjoyable it was, and its ability to help people think of and frame their particular architectural preferences. Volunteers for the study were solicited from multiple sources; a total of 12 volunteers participated in this study.¹ The volunteers were not designers or architects by profession. The respondents were 66% female; all but two were born between 1965 and 1982, the two older subjects were born in 1942 and in 1947. The study was conducted in an MIT research office, and user sessions averaged about an hour in overall length.

5.1.1 Protocol

The study had three parts. First, users completed a short 7-item questionnaire to specify their familiarity with architectural design (i.e., have they ever worked with an architect or looked at floor plans), and also their level of proficiency with computers: what types of programs they use, and how often they use those programs. (Appendix 1, A1.2) Next, the users worked through each of the five exercises: Component Assembly, Option Checklist, Activity Sequencer, Packing Metaphor, and Diagramming. After each exercise, the user had a chance to rate the exercise. Rating was done using a common set of assertion statements with which the user could either agree or disagree.² In addition to these ratings, the user was given the opportunity to make general comments immediately after each exercise. Upon completion of the exercises, the volunteer finished up by answering a few general comparison questions about the exercises and was given an opportunity to make any other comments about each or any of them.

For each of the exercises, users were given standardized instructions verbally. The order in which the exercises were presented was varied. Users were given the opportunity to ask questions, of the

“Frame-Arrays” and Chapter 26 “Language-Frames”. Also F.C. Bartlett, Roger Schank

8. The final chapter of *The Place of Houses* by Charles Moore, pp 241-266.
9. Our initial diagramming tool allowed zone overlapping and the flexible sizing and placing of zones. Initial testing showed that people didn't really know where to start, and that the overlaps were ambiguous. Also, people's diagrams tended to focus on the more literal spatial organization of rooms and not their abstract relationships. This was problematic because the spatial organization of diagrams is not meaningful to the search algorithm which focuses more generally on room-to-room connections and room functions. To make the diagramming simpler and the diagrams more meaningful, the prototyped exercise constrains all zones into a pre-drawn grid. This simplifies the diagramming process because people do not have to worry about where to put the zones, which in turn eliminates the tendency for users to be overly concerned with literal organization. The grid also makes it easier to start because it's no longer an empty page, in addition to the fact that the user has fewer elements to worry about placing.

author, about the exercises if any clarification was necessary. During the exercises, users were permitted to ask for additional clarification if necessary. Apart from answering these specific questions, I avoided any verbal communication with the participant during the exercise. As the interviewer, I was present throughout the exercises, seated facing the users at the table upon which they worked. For each session, audiovisual information was recorded for further analysis.

The volunteers came from a variety of backgrounds and experiences. One was a retired head librarian, another worked for IBM, and another was a student finishing up her Masters degree at Emerson College. None of the volunteers had ever worked with an architect to build or remodel a home. Seven of the 12 participants identified themselves as being comfortable with floor plans; the others were only vaguely familiar with them. All of the users described themselves as either regular users (9) or power users (3) of computers. In terms of the applications used, the 9 regular users tended to follow a routine of computer use and the 3 power users tended to actively seek out new programs or functionality. All of the users were familiar with surfing the Web, using Google, using Google earth, checking email, using Microsoft Word and using Microsoft PowerPoint. Only a few users were experienced with computer programming software, CAD software, or graphics software, and none were proficient or expert users of those types of software.

Prior to the evaluations there was a concern that the users would be influenced by visual examples given, particularly with the diagramming task, but there was no observed evidence that this was the case. To help circumvent the possibility of said influence, the examples were not left in front of people during the actual interviews. In the following sections, the salient findings of the studies will be summarized, first with the component-based exercise and then with the conceptual exercises.

5.1.2 Jeffrey

Jeffrey, one of the users who volunteered for the study, required a somewhat different protocol because he was legally blind. Because he was unable to see the exercise materials, each of the exercises

was done cooperatively, where the options and arrangements were described verbally by the author. In addition, Jeffrey was unable to go through the Component Assembly exercise because he couldn't see the various images that were presented and there was no benefit in trying to verbalize all of the meaning those images contained. However, his perspective of the conceptual exercises was, as whole, illuminating because of his unique relation to space. The most salient room qualities for him were warmth and quietness; they were not visual things. And he tended to focus more on the intricate details of his movement through space. For example, in putting together the Activity Sequence of his morning routine, and thinking about moving through a doorway, he stated "I have thresholds in the doorways, I hate them. I walk around and anything that's on the floor bothers me." Conceptually, the pathways that he makes through space are his most dominant concern. "I have a unique problem: I'm blind. Wanna see me carry a bowl of soup from the kitchen to the living room? I don't have an eat-in kitchen, it's a little galley kitchen. So I'm looking for the shortest walking path between the kitchen and the table." With that said, it was encouraging to observe how salient the conceptual structure of each of the exercises was for him, even without the visual cues from the workspaces and various pieces. In a way his unique session was one of the more important ones because it helped confirm that the underlying structure of the various tasks was clear and approachable, even without the benefit of sight.

5.2 Component-based exercise, Findings

Overall, the volunteers found the Component Assembly exercise detailed in Chapter 4 to be easy to understand and work through. Even though several users did not consider themselves to be comfortable with floor plans, the concept of a floor plan view was very obvious for everyone, and only a few of the volunteers asked for any clarification about the specific items shown (i.e. "is that the fridge?") or how rooms connected together. Because efforts had been made to describe the options in a number of ways, some of this clarity may have come from the variety of different representations shown in the library cards for each space. (Figure 3.8, Appendix 1, A1.3)

So, we asked people what representations were most useful to them during the selection process. Most users said that the plan view, or the plan view in combination with the 3D view or the rendering, was most useful. And two users thought the floor plan diagrams were most useful – only three users thought non-plan representations, like the 3D view or the renderings, were more useful than the plans themselves. Still, having the entire variety of options seemed to be useful for people, even as some subset of those representations was clearly more important for everyone. As one user, Megan, noted: “I rarely looked at the photos but then when I had [options] that didn’t have the photos, I missed them a lot, I wanted them back.”³

In general, going through the component assembly process room by room seemed to be an effective approach to simplifying the selection process, because people could separate the rooms conceptually. But, there were a few issues with this separation: mostly with the “bigger picture” of the overall plan’s design being something that the users didn’t give much thought to or engage in. Instead, they tended to focus upon the rooms in isolation. Take for example, Marisa, a volunteer who is preparing to relocate for medical school, who noted: “the only thing I thought about between the rooms was closet space... other than that I just thought about the rooms kind of independently.” Another user, Thea, preferred to look at the master bedroom and master bathroom at the same time, because for her, they were conceptually part of the same private zone and difficult to select separately. Closets were a very popular consideration in private spaces like the bedrooms and bathrooms, and several users verbalized that they were important. In the more public spaces, a significant number of users were trying to select rooms that had a sense of “openness”. But surprisingly, highly popular layouts did not emerge from the options given for any of the rooms, with the exception of the living room where one configuration was selected five times.⁴

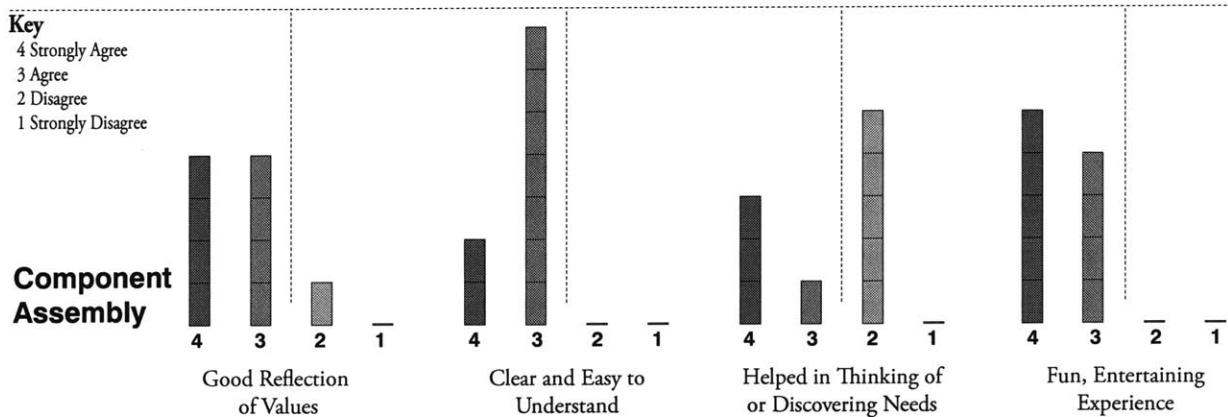


Figure 5.1 Component Assembly Ranks
Users were asked to rank each of the exercises according to the criteria here. These are the responses for the component-based exercise

Everyone liked the exercise, and all but one agreed that the plan that they made was a reflection of their values. (Figure 5.1) But as it turns out, and this is fundamentally important to this analysis, the real question is *which* values were being represented by their plan selections? Because even while the exercise was approachable and gave the people a sense of accomplishment, there were several potential problems observed, the first of which was a tendency to avoid revisiting any of the spaces that had been selected.

Essentially, people tended to avoid going back and changing a room once they had picked it. For example, if a user selected a bedroom, and then selected a bathroom that wouldn't fit with that bedroom either due to size or doorway alignment, they would tend to assume the bedroom to be a given and go with another bathroom that fit. This was true even though each user was encouraged to go back and revise any selection at any time, if they so desired. Only one user, Liv, went back to make significant changes, and two others made just a single revision to one space that was already selected. Apart from those three users, the rest of the group made no revisions at all to their original selections. As Zahra, an MIT graduate, noted: "I guess I made some choices later because I was constrained by earlier choices where I didn't know what I was doing." This finding was a bit of a surprise because it was expected that the iterative process of refining selections would be one of the more entertaining and useful aspects of this particular exercise.

There could be many reasons that revisions didn't tend to happen. First and most obvious is the possibility that because people were conceiving of rooms separately and not in terms of the overall plan,

there was no need to go back and revisit anything according to some overlying concern because that concern simply didn't exist. So instead most users seemed to use the question, "what fits with what I have?" as additional criteria to help make their selections. Secondly, the selection process for the rooms was highly comparative in nature and took each of the users quite a bit of time to complete.⁵ So if they had invested time in the choice, and already had a sense that it was the best choice, why then should they revisit it? This issue hints at the possibility of a more fundamental problem, for which other evidence was observed, which is simply that the selections made were highly context-specific.

Figure 5.1 shows that most users didn't think this exercise helped them think of or discover their own needs. One possible reason is that the selections that people made in this exercise were highly contextualized by the design of the room arrangements and the specifics of the representations. So, while people did feel that they had picked the room they valued the most, this value judgment was made based upon straightforward things like how the furniture was depicted, or for example, how a table was shaped. Kevin, a 35 year old, noted that he liked one of the living room options because of the shape of the coffee table that it depicted. Apart from Kevin, some users externalized their thoughts about the selection process while working through the exercise, revealing a bit of their conceptual speculation. The following are a few examples:

"I kind of like having no table. I don't sit at the table in the kitchen a lot. I guess I could, I probably should but, well, that's a big table. I mean, do I need to be that literal? I mean that is a big table. I would never need to sit there, I would sit there all alone and that would just be terrible. There's room for what 8 people there? I don't have 8 people that I want to talk to all at once. No! Get out of my house!"

Brittany, who went on to choose a kitchen with no table

"I really don't want the TV at the end of my bed, but I kind of like the layout"

Brittany (she ended up accepting the TV)

“I’m not a big Sofa person”

John, a retired librarian, reflecting on the living room furniture in his selection

In reflecting on her own design, Zahra described further her selection process: “well, there were a few items that I knew for sure I had to have ... like the closet in the bedroom ... but then there are other features where I wasn’t able to distinguish very clearly what my needs were, so all of the choices seemed the same”. When considered, Zahra’s comment illuminates two issues, first that her selection process was based mostly on literal room features, and secondly that the only decisions she could make easily were the ones for which she’d identified a need ahead of time. This observation points to the third fundamental problem that this evaluation uncovered: that working with the more literal representations didn’t support a varied conceptualization of the space, or the identification of new needs or values by the user while *doing* the exercise.

As Zahra’s comments suggest, the people who had the easiest time with this exercise were those who had spent more time reflecting upon their spatial needs beforehand. A good example is Thea, who is currently in the process of renovating a house in New Haven, Connecticut, with her husband. Of her selection process, she describes that “we’ve spent a lot of time thinking about the way we like to live, and having this area (point to the living room) open, so it wasn’t that hard [to decide].” But everyone appeared to have at least one or two high-level goals. So the pre-framed needs that made this process doable came from some sort of earlier reflection process, and were in all cases a reaction to an immediately observable *problem*. It’s not a surprise that for 12 users who live in the cities of Boston and Cambridge, MA, closets and large open spaces were the most sought out features, because most, if not all, of them have been living in and adapting to units or houses that are small and/or old. But really obvious problems should not be the only way spatial needs are framed.

Ultimately, the room selection process, when made in the context of the component exercise, was simply about choosing the best from a number of available options, by using the information that was provided; as previously noted, people were engaging in a

specific comparative analysis. With the information being literal and configuration-specific, it is not really a surprise that the exercise did not tend to cause any further conceptualization of the space, but it is important to point out because it's the most critical shortcoming of the exercise. Are people really arriving at the best, most livable selections for their home, and what decisions are they actually making? As our findings with the conceptual exercises suggest, far more valuable preferences can be elicited through constructive tasks that encourage reflection inveterately.

5.3 Conceptual Exercises, Findings

The conceptual exercises detailed in Chapter 3 were evaluated to determine how well they help people frame new problems or needs in terms of their home preferences, to get a sense of their educational value, to see how easy they were to both understand and work through, and to find out how entertaining they were. The findings presented in this section are structured according to each of these areas of inquiry, starting with problem framing. Within each area of inquiry, findings specific to each of the exercises are described.

Exercise Summary:

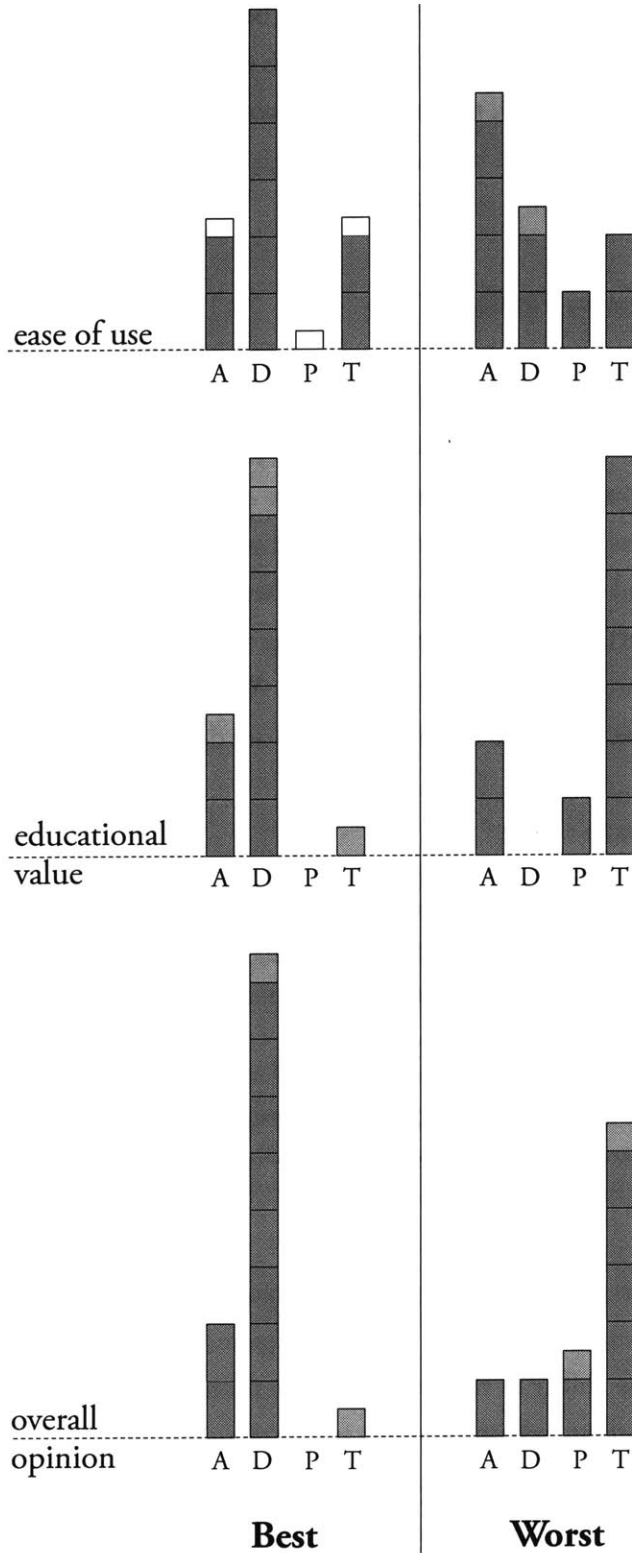
- *Text-Based Checklist*: selecting one option from a checklist of text descriptions, broken up room-by room
- *Role-Playing Metaphor*: users imagine they packing up *items* for an upcoming move as a way to think about their space needs
- *Activity Sequencer*: users list out the *activities* within their morning routine as a way to think about their space needs
- *Floor Plan Diagram*: users construct an abstract *diagram* that describes their space needs.

Users were given the opportunity to comparatively rank the conceptual interfaces, the results are shown in Figure 5.2 on the following page. The ability of the conceptual exercises to help people think of and frame their individual needs is central to the system's success in letting people conceptualize of their spatial preferences. For this reason, problem framing was a central concern of these

Figure 5.2 Conceptual Exercises Comparative Ranks
 Users were asked to rank each of the conceptual exercises from best to worst in terms of ease of use, educational value, and most liked overall.

Key
 A activity sequencer
 D floor plan diagram
 P packing metaphor
 T Checklist Select

□ 0.3 pt (3 way tie)
 ▒ 0.5 pt (2 way tie)
 ■ 1 point



evaluations. As will be shown, the difference between helping people think of things and being educational is a fuzzy one, but in terms of educational value, it's important to consider not only the learning that may have been done about needs, but about the actual exercises as well. Ease of use was analyzed to determine how well people could understand and also *do* the exercises; the concern here was with identifying any potential barriers to users getting started and working through the exercises. In the context of a web-based implementation, approachability is tremendously important because users aren't likely to hang around if they can't figure things out. And along those same lines, the entertainment value of the exercises was evaluated to get a sense of just how fun they were. As will be discussed, being fun is important for the conceptual interfaces because it helps provide an immediately tangible benefit, and also encourages users to spend more time conceptualizing.

5.3.1 Problem Framing (Figure 5.3)

Each of the conceptual exercises described herein incorporated description and reflection stages to help users think about needs and values. The goal behind this bipartite structure was to utilize the process of first constructing a description of something as a way to then reflect upon the implications of that specific something. There is however, a notable exception to this, and that is the Checklist Specification exercise, the process for which more closely mirrors the component assembly exercise detailed in the previous section. Here, it was observed that users were simply selecting the choice that seemed best, in one step. Resultantly, the findings for the checklist exercise tend to take a departure from those for the others. As Marisa describes, the Activity Sequencer “makes you think about what you need whereas [with the Checklist] you don't really think about functionality that much, you just think about what you think would be good.” This contrast was expected; in fact, our primary intention with the provision of a checklist task was to compare that task with the other exercises, because text-based option selection is the de facto standard for existing home search functions as of 2007. ⁶

Compared to the Component Assembly and to the Checklist

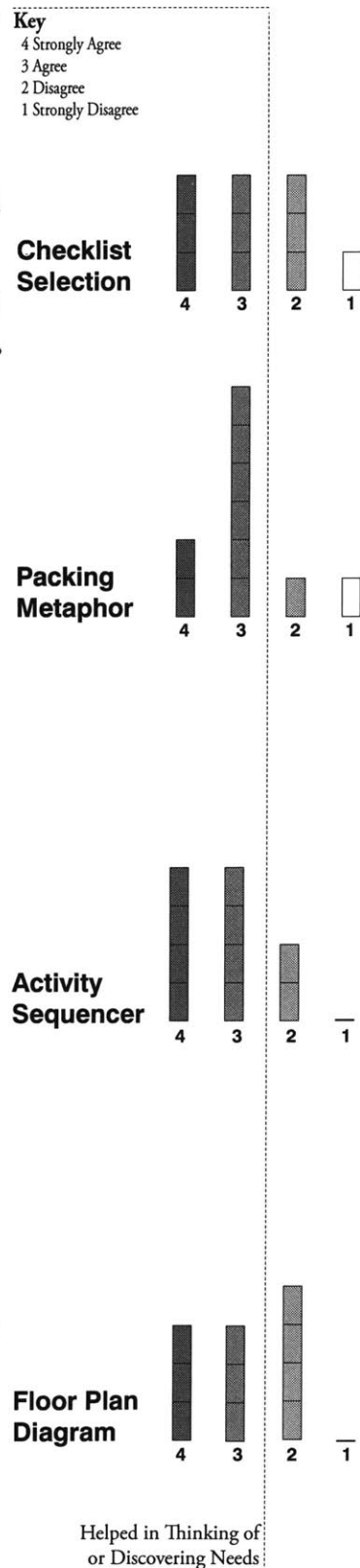


Figure 5.3 Conceptual Exercises Ranked by Ability to Frame Problems
 User rankings of how well each of the exercises allow them to discover their needs

Specification, the other exercises allowed more significant and more varied discovery of spatial needs. When asked if the three more conceptual exercises helped them to think about their home needs in ways that the Component Assembly and checklist did not, all but one of the users agreed. As Marisa noted, “they made me think of the bigger picture.” Another user named Alison explained: “the exercises made me consider what I really value in a space, and made me think about how my current living situation differs from my ideal situation.” For each of the more conceptual tasks, the aforementioned description stage focused on the user’s current living situation, followed by a speculative reflection stage where users then identified their preferences. This allowed them to think about both the good and the bad things about their current situation, as opposed to the Component Assembly, which only seemed to be evocative of the profoundly bad things if it was evocative at all.

For several of the users, the Activity Sequencer helped them to recognize how much time was spent in certain rooms and what they do those rooms, and also helped them discover inefficiencies in their spatial configurations, given the activity being described. After completing her morning’s description, Thea described her thoughts about spatial revisions that would help make her activities more efficient: “[I could try] combining ‘Waking Up’ with ‘Getting Dressed’ [into one room] so that I don’t have to move through doorways or hallways... you can just, you know, move through a room”. While working through her sequence, Brittany thought about where “Eating” happens in the morning: “I’m at my office desk! That’s where I sit and eat. And I’m doing something at the computer ... which seems really weird under eating, I realize, but that’s really what I’m doing, working and eating are really kind of synonymous.” Building up these activity-based descriptions made it much simpler for people to describe items that were important to have in the room, or room qualities. As such, it was observed that for many people, having lots of natural light was a particularly important room attribute that emerged when they thought about waking up. But the exercise also helped people reconnect with unique and important things about themselves. When asked why he listed “Knick Knacks” under his “Reading” activity, John noted: “I’m very much into

collectibles. And I have a lot, my house is just full of them and that's part of my psyche. Without those, I could not go from showering to reading because I always stop and look at them". Later, when selecting his living room in the Component Assembly, he referred to this specific discovery to help select a room layout.

Similarly, the Role-Playing Metaphor of packing up a room encouraged people to think about the relative importance of the various things they keep in that room, and also how items changed the room. John, in working on the kitchen, noted that the packing activity made him "think just how important the telephone is, and [that] pictures are meaningful." While packing up his living room, Jack, who works for IBM and participated in the exercises with his wife Jane, said: "I never thought about how many seats and chairs we have in the house." However, while the Activity Sequencer was popular, not everybody thought the packing metaphor was tremendously useful. Zahra noted: "the things that I take with me in the room, for the bathroom at least, are not going to change really". This may have been because the bathroom tends to have a more constrictive and standard layout, but there were several other users who felt like reflecting on their list of stuff was not very useful when it came to identifying preferences. In these cases, the connection between their description of items and their reflection on preferences was a fuzzy one for users; they had to make a bit of a jump. Still, as Thea noted: "it would be helpful to do that for every room".

For the diagramming task, users were asked to first diagram one floor of their current living space, and then make one to three improvements. So again, the process was one of description, followed by reflection. This did help users frame problem spots, by thinking through how the space was structured. As an alternative, some users were told they could start from scratch; all but one of the users who were given this option did so. ⁷ This alternative was useful because for some of the first few volunteers, their current living situation was very restrictive and they would often either completely start over during the reflection stage, or make only small adjustments, like changing the function of a certain room.

Because the task was so constructive, there emerged much more of what Schon described as Reflection-In-Action, and some users

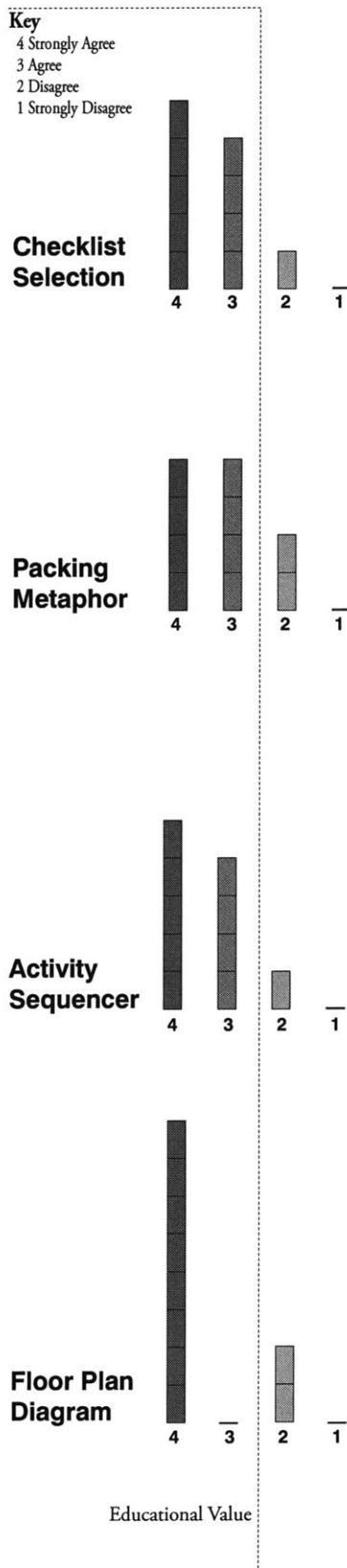


Figure 5.4 Conceptual Exercises Ranked by Educational Value
 User rankings of the educational value of each of the exercises

externalized that even while starting from scratch, they were thinking of places they had known or lived in. For example, Brittany reflects: “I’m also thinking about my childhood home. My parents had this really great house so that’s a good, a really good model, and when you walked in ... it was just open.” Additionally, in Brittany’s case, there were multiple sources reflected upon: “this is coming from the idea of my grandparent’s old house; where there was a giant bathroom between our bedrooms ... I thought that was really nice.” Her ultimate diagram was the combination of the two sources. Other users like Jeffrey, who started with his current situation, were able to highlight specific areas of need: “I have no second exit ... that bothers me”, in addition to more high level concepts: “the practical reason for design change in my apartment is to minimize the walking path.” In each of these cases, the diagramming exercise elicited thoughts about spatial preferences, simply by letting users build something.

5.3.2 Educational Value (Figure 5.4)

In addition to examining how well these exercises allowed people to discover their needs, I was curious if people *learned* anything new by working through the exercises. The ratings in Figure 5.4 are quite positive for all exercises but not particularly useful, comparatively. The users were divided fairly evenly when asked if the conceptual exercises allowed them to learn something new about their preferences: only 55% agreed that this was the case. As it turned out, it was very difficult to get a sense of whether the communicated preferences were new concepts or simply ideas remembered.⁷ Furthermore, it was probably difficult for the users to think of new things that they had learned while doing the exercises, because this knowledge may have been tacit or subtle. And there is also the case of meta-knowledge, knowledge gained about the exercises themselves and their purposes. Several people, like John and Jack, described that the Activity Sequencer allowed them to think about their space use in a way that they had never thought about before, which suggests that they learned a new method for describing their needs, even if they didn’t necessary learn anything new about their needs.

Lastly, it is difficult to separate the process of learning from the

process of discovering. Again, all of the users felt like the exercises allowed them to identify and describe their spatial needs. This was not necessarily new knowledge, nor does it need to be, but perhaps it was being organized in a new way. Perhaps discovery is an educational phenomenon, even if it is rediscovery.

There was some evidence of tacit learning when people talked about their experiences; for example, comparisons drawn between the exercises themselves. Several users, when describing their efforts to work through the Checklist, noted that the specification seemed inappropriately flexible. Zahra notes: “it’s hard to visualize [the Checklist] because I’m not constrained on anything; here (with the Component Assembly) you’re completely focused on what you’re giving up.” Another user, Alison, reflects similarly, “I thought I could choose better with that one (pointing to the Components) because I could see what it looked like. I’m not even sure if what I picked [on the Checklist] is the same.”

And in other sessions, users said or did some surprising things. Consider this brief exchange with Liv, a recent college graduate, in discussing her diagram:

Liv: “it was a little restrictive because I’m thinking to scale when I know I shouldn’t be”

Giles: “So how did you know you shouldn’t be?”

Liv: “well, just because of the grid”

Here, Liv had inferred knowledge about what she should be describing simply by observing the structure of the Diagram’s workspace. After pilot studies, the grid was incorporated into the design of the exercise to help make the process more approachable, but it is also intended to help the system prevent people from representing space too literally to be of any use as a query. This tension between constructing something and having it be searchable is unique to the diagramming task, and Liv had discovered this. Also, several of the users made observations about how the tasks themselves could be related. Thea described that it would be useful to do the packing metaphor for each of the rooms in her diagram; Marisa noted that the exercises could be linked together to help complete

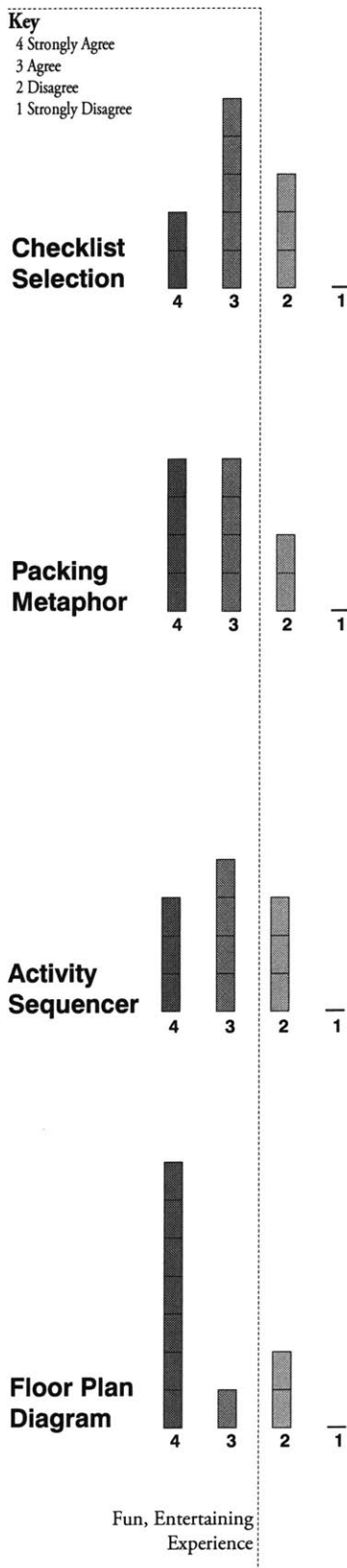


Figure 5.5 Conceptual Exercises Ranked by Entertainment Value
User rankings of how fun each of the exercises was

her description. This sort of thinking, of applying ideas about how the tasks can be organized and structured, implies that aspects of the process were educational enough to engage people into the thoughtful evaluation of that process.

5.3.3 Fun for people? (Figure 5.5)

The entertainment value of these conceptual interfaces is crucial, for both practical and technical reasons. Speaking practically, it is important that the conceptual query interfaces be enjoyable to keep people engaged and interested in the task at hand. This is particularly true for adults, who, as was previously described, require the perception of some benefit associated with doing something to be willing to spend the time to do that something. For users where the benefits of the constructive tasks are not immediately discernable, the potential to have *fun* with the search is the immediate benefit that keeps them engaged. Also, it seems probable that having an entertaining interface is a good way to get people to spend more time not only to complete their query but to work to refine it. This is a technical reason to try and achieve interfaces that are actually fun, because for the underlying search process, more complete and more refined queries mean better search results, which leads to a better system overall.

Though it's hard to evaluate in any quantifiable way, it seems that with the exception of the Checklist exercise, all of the exercises that were presented were quite enjoyable for people. Users ranked the exercises as fun, and additionally, there was plenty of observed evidence that everyone was, to put it simply, having a good time. For example, 10 out of 12 people opted to finish up their exercises and have the session take longer. So, while sessions were designed to last 30 minutes, most of them lasted an hour and some even more. In several cases, the users were asked to stop working on an exercise that they were immersed within, so that they could move on to the next one; these users tended to be both surprised at the amount of time that had passed and to want to finish up what they were doing. Some comments that were made during the activities:

“I’d like to sit here for a couple of hours by myself and work on it”

– John, on the Diagramming Task

“It’s like monopoly... this is fun!” – Jane, on the Activity Sequencer

“I just think it’s so cool ... you guys should create a game” – Jane,
on the Diagramming Task

“This is great! This is so much fun. More people should have
things that are this fun to do, this makes me want to do all of this”,
Brittany, on the Packing Metaphor

Overall, the Diagramming task was the most engaging and the most fun for people. In general, users tended to favor one exercise over the others, sometimes extremely. So from the group of users there emerged people who were, for example, activity sequencers, and people who were diagrammers. It seems likely that the different exercises would be well suited to different problem solving abilities of people. For example, Alison, who described herself as a very visual thinker, favored the Diagramming over the more narrative Activity Sequencer. But Zahra was the exact opposite. The Checklist, being textual and straightforward, and also containing options that were generally unclear, was found to be the least enjoyable, and some people found the Role-Playing Metaphor to be smaller in scope and not as engaging when compared to the other exercises.

Jack and Jane, who worked on the exercises together as a couple, discovered within the structure of the exercises both an interesting space for communication and a way to have fun with each other. Here is an example, from their conversation during the construction of the diagram:

Jane: “I’d like a chair right there...”

Jack: “You like chairs everywhere!”

Jane: (laughs) “Yea, why not, it’s a nice sitting area, like in a hotel, you know, they have the lobby area. It’s a dream.”

The exchanges that Jack and Jane made throughout the course of their session point to an interesting dimension of the conceptual interfaces: that they could be made to be interactive, and to allow multiple people, of different relations, to build queries together at the same time. The final chapter will discuss this possibility in more

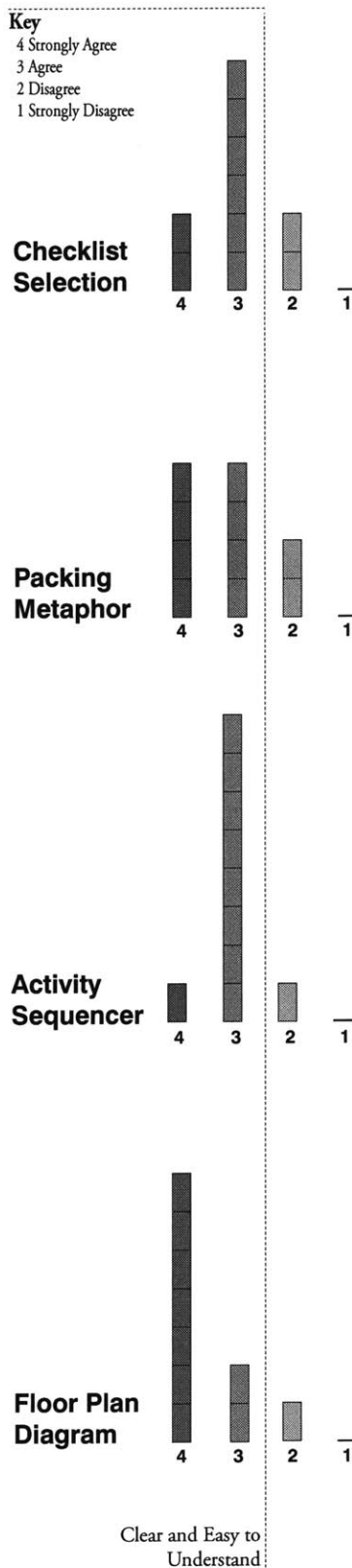


Figure 5.6 Conceptual Exercises Ranked by Ease of Use
 User rankings of how easy each of the exercises was to understand and work through

detail. Beyond that, it seems likely at the exercises were deemed to be enjoyable because they were clear and because they allowed people to both work towards and sustain identified goals. Since the exercises were constructive and engaging, game-like experiences simply happened.

5.3.4 Ease of Use (Figure 5.6)

It goes without saying that these exercises needed to be easy to use; so that people could quickly engage in a constructive process of discovery without being frustrated by a complicated process. But it became apparent through observation that there were really two types of easy: the clarity of the instructions versus the approachability of the choices the exercises forced people to make. The Checklist is a good example, because the rules couldn't be simpler, but the task was still a difficult one. This is because the options were difficult to distinguish and to visualize. Interestingly, it was discovered that people who did the Checklist as the first exercise actually had more trouble with it. As it turns out, this is because the people who had done either the Component Assembly or the other conceptual exercises ahead of time were able to refer to those preceding exercises to help visualize the Checklist options and to make their choices.⁹ Brittany, who did the Checklist first, exclaimed: "this is so complicated, let me know if I'm putting too much thought into this!"

For all of the conceptual exercises that involved tagging, the semantic definitions that were included on the attribute tags were useful to the users in clarifying the meaning of those tags. This finding supports the notion that by providing the definitions to the user ahead of time, we may be able to clarify the functionality of the search operation itself. Looking at the "Efficient" attribute tag during the Activity Sequencer, Jane commented "oh and this, based upon the description, seems like it makes sense."¹⁰

For many, the Activity Sequencer was, while interesting, a little difficult, but most people still seemed to find it engaging. Brittany noted: "it's hard to qualify what I'm doing, I might be sitting on the floor with the cat, and it may be a little out of order ... I might just get up and work for a while, not even eat." And John described that

“it was hard because I had to think of what really means something to me when I get up in the morning and I’ve never really thought about it that way”, but he was very happy with his discoveries.

Four of the users also found the grid in the Diagramming Task to be a bit limiting to work with; but only because they wanted to have a bit more freedom in defining room boundaries. For reasons previously described, this had been disallowed. Still, all told, the easiness of the Diagramming Task itself was one of the most unexpected findings of these evaluations. Going into the evaluations, the assumption was made that the task would be quite difficult, to the extent that people may not even be able to do it. But observations revealed quite the contrary result: that it was very approachable, quite fun, and in many ways easier than trying to, for example, sequence together a collection of activities because the diagram was more directly representative of space. But of course, while the ease of the diagramming task was a pleasant surprise, there are other, important considerations that challenge the use of this exercise in particular.

5.3.5.1 Pros and Cons: Text-Based Checklists

As an exercise, the Checklists were easy to understand, but difficult to work with. Without any visual information, users found it difficult to differentiate the choices given, unless they referenced other exercises or had strong convictions about particular features. As described in section 4.2.1, the exercise would have scalability issues if incorporated as a system interface. And there was no evidence that people discovered any needs by working on this exercise. For these reasons, the checklists were not incorporated into the prototype.

5.3.5.2 Pros and Cons: Role Playing Metaphors

For some users the packing metaphor was not particularly enjoyable as an exercise. And a few users noted that the connection between listing items and then identifying valued room attributes was not meaningful. These findings suggest that this exercise might not be good for everyone, or that it might be more useful only in specific rooms.

However, for some users it was very useful to think about all of their items as a way to begin thinking about what they'd need from a space. Several users also found that making the list of items was a good way to remember about all of their possessions and even to prune out unnecessary or unwanted possessions

5.3.5.3 Pros and Cons: Activity Sequencer

The Activity Sequencer was, for several users, the most difficult exercise and required more explanation than any of the other exercises. Also, users found that the morning routine isn't always a distinctly sequential set of activities, and that it may vary from day-to-day, which suggests that even if thought-provoking, the specific sequence given might not be wholly accurate. This isn't a significant problem because the attribute list that the sequence generates is what's significant to the search.

Also, it was found to be the most useful of all exercises in allowing people to discover needs and values, and it was evident from the sessions that even though the task was initially more difficult than the others, it was very useful. Additionally, the activity sequencer encourages people to think in detail about multiple rooms and the way they flow together, in addition to how those individual rooms are utilized. This means that more query information is generated by this interface than the others, and therefore one of the more effective interfaces at getting good results. A more detailed query gives the system more to go on. This type of specificity will be explored further in Chapter 6.

5.3.5.4 Pros and Cons: *Floor Plan Diagrams*

People found the diagramming to be the most enjoyable task. And, the diagramming was unique among all of the exercises in the amount of improvisation that took place. In many cases, users bent or even broke the rules to reach their final designs. Rules can, of course, be enforced more strictly in a computational implementation, but the specific nature of the ways that people strayed is very informative and should not be overlooked.

Earlier in this Chapter, it was briefly described how one user, Liv, intuited that this exercise was not intended to allow people to describe the relative sizes of rooms. But, three of the users did so anyway, by using the “open connection” element to effectively merge cells together. And fascinatingly, each of these users employed the same convention to indicate that the cells that were now open to each other were in fact the same space: they laid the room label over top of the open connector to span the multiple zones. According to the rules that were given to the users, this was not allowed; yet, a significant number of them did it, just the same. For the query interface itself, it would be simple enough to allow this merging of cells to describe the relative size of rooms, but in terms of the search operation, determining a match based on size is far more difficult; the system would either have to rely on proportional comparisons or standardized ranges. Another common improvisation was with the “Hallway” tag, which was intended to be used as a room label and therefore placed within a cell: multiple users placed it between cells, just like a doorway connector. So, some people seemed to conceive of Hallways as a connection, as a conduit, rather than an autonomous space.

These improvisations highlight one of the fundamental issues with the Diagramming task: the user’s idea of what they are specifying not matching up with the system’s idea of what they are specifying. From these evaluations, it was apparent that some users expected that size would be meaningful, and also that specific orientations would be meaningful. These criteria will be considered in the detailed system overview of the final chapter.

The issue of unmet expectations is made further problematic by the fact that people also tended to make mistakes during this exercise. In making connections between rooms, a few users either overlooked some rooms or made a strange chain of rooms. Zahra, for example, connected rooms together into isolated pairs without tying the overall plan together. And Brittany had a bedroom that could only be entered through a bathroom. Because the underlying search process allows for design criticism to be added to the construction process, through detailed relevancy reporting, initial correctness might not be critical. These user exercises only looked at the first

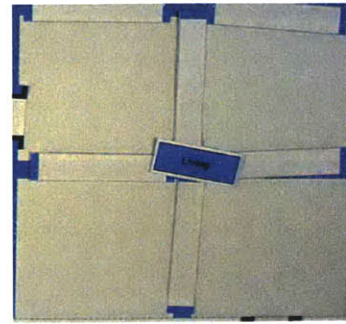


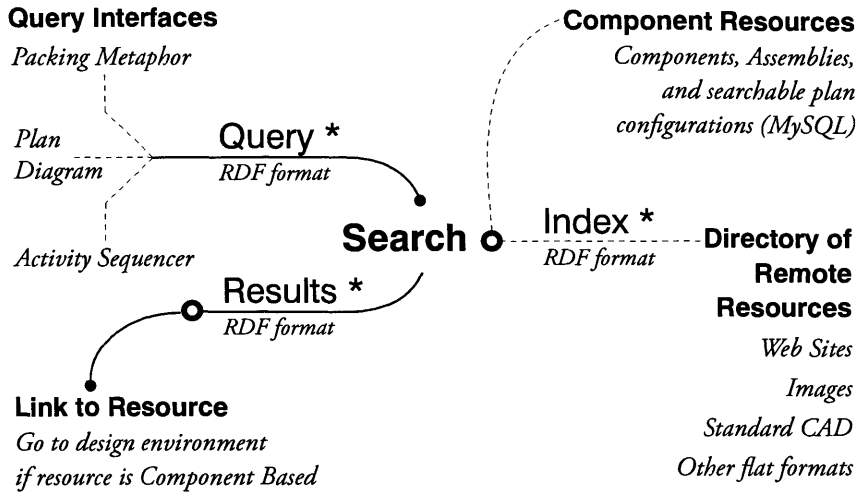
Figure 5.7 Improvised Bridging Technique for Making Larger Rooms
Here, the user wants a larger Living room, so they've placed the tag to "connect" the cells.
3 Users employed this technique.

step of the potentially iterative process of searching; only the initial query building. Whether or not a fragmented or peculiar diagram would be able to output an effective query remains to be seen, but it is important to note that with the diagrams, the process was not only more constructive than the other exercises, but also led to a higher incidence of errors. ¹¹

1. Solicitation was done via the MIT House_n volunteers mailing list, the Harvard Technology in Education mailing list, and a Web posting for volunteers on craigslist (<http://boston.craigslist.com>)
2. See Appendix 1 for the complete questionnaire including the exercise rating
3. Some of the rooms presented in the Component library were missing renderings, simply because there wasn't time to get them all generated prior to the study. This user was referring to those rooms with no renderings, which she called photos.
4. From descriptions they'd given, 9 of the 12 users wanted a sense of "openness" in their plans. The popular living room was option #9, with 4 chairs and one sofa around a large rectangular table alongside an open study area separated by a shelving/wall unit.
5. From observations, the selection of a single room took about a minute. The kitchen tended to take the longest; this may have been because most people started there so it was the first one they'd looked at.
6. The "Design Your Home" application hosted by the Toll Brothers is a good example of text-based search. The query options given in this example are State (region), Number of Rooms, Square Footage, Number of Bathrooms, and Garage, followed by a Style selection if any options come back.
7. The ability for people to start working on the diagram from scratch was discovered when the fifth user came in. This came as a bit of a surprise because I'd anticipated people would have more trouble with the diagram. For the user in question, Kevin, his current plan was bad enough that he simply didn't want to start with it, so he did his own thing. Our discussion afterwards uncovered that he felt like start with his current situation would constrain him. Subsequent users were given the option to start from scratch.
8. For me, the separation between discovery and learning is not a clear one; I would argue that all discovery or remembrance is learning. In theory, ideas are intransigent, perishing things... thinking of them in some new context means that new information about the meaning of the idea itself, however insignificant, has been recorded.

9. I first noticed this was happening with Megan, she was the first user that didn't have the option selection first and she was clearly trying to find the option that matched her Component Assembly design. Because the text descriptions are so general and difficult to visualize, it seems natural that the process would be a porous one, where people are conceptualizing based upon immediate experiences and visual information.
10. The session with Marisa uncovered an issue with the semantic attribute tags that I was expecting to see more of, simply that the attributes provided (see Figure 4.2) were not numerous enough. She wanted more options. None of the other users had given this indication, so it seems likely that even in the number of attribute tags needed to be expanded, even doubled or tripled, we'd still be dealing with a manageable set.
11. In the case of unspecified connections, it seems likely that the result would simply be a weaker search, because the query interface would not be able to report relevancy limitations from broken connections, it has no mechanism for inferring these details.

Figure 6.1 Search System Structure
Showing multiple interfaces and multiple searchable representations



** RDF for all transported data is in the same format*

6 Search System Prototype

6.1 Overview

This chapter summarizes a proposal for a consumer-based design search system. ¹ For the underlying component-based system, only the data structure and not the interface was developed. While this work is centered upon the search, some details are here provided for a more complete and multifunctional system. This system is a search tool that allows users to design queries, which are then used to search for home solutions. When the search results come back to the user, they may contain either component-based representations or any number of flat representations that have been adequately tagged with indexes. In this system, the process of searching allows for the initial discovery and specification of preferences through conceptual designs, and for the selection of a buildable design solution through the matching algorithms. But further, as we will see in cases where the search results are of a component-based structure, the system allows for the more detailed adjustment of the design solution within the limits of that component representation. So in the context of a component-based solution, the process is divided into two distinct stages where the search operation allows the user to arrive at a preferable pre-designed configuration as a starting point, and then to further tweak the design according to the rules of the component

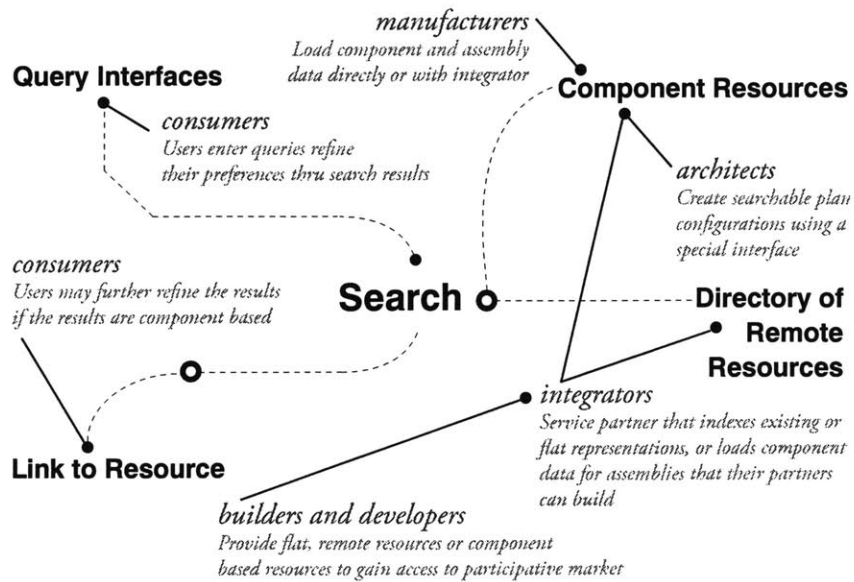


Figure 6.2 User Roles within the System
 The key players and how they interface with the system detailed in Figure 6.1

system itself. In the case of other, non-configurable representations, the designing ends when the search itself ends.

The front-end to the system is the collection of conceptual search interfaces based upon those detailed in Chapter 4 and further refined based upon the results of user evaluations. (Figure 6.1) The Checklist interface was not included (see Chapter 5 for details). The interfaces are web-based and can search various types of web-accessible resources; the interfaces can function either individually or collectively to allow for more refined searches. The back-end of the system, to which the query interfaces submit their requests, is a relational database that either points to or stores design solutions from various content providers, depending on the type of content. These design solutions are linked directly to the results listing of the query interface. As far as the search is concerned, the resources could be anything from flat, static images like Jpegs to complex Component systems that are attached to pre-fabricated building assemblies. The differences between the various searchable representations will be detailed in the *Results Listing* section of this chapter.

6.2 Users

There are 3 basic classes of users within the system: consumers, content providers, and integrators. (Figure 6.2) The consumers are simply the users that come to the site to search for home

preferences and solutions. They can be expert, non-expert, or just curious – no assumptions are made about their reasons for arriving at the site, but it is assumed that they will engage in a search. The consumers create, through interaction with the query interfaces, search requests for the system. To gain access to searchable content, the system utilizes industry partners as content providers that either offer indexed, pre-designed solutions or contribute to a centralized library of components and assemblies that support the provision of component-based solutions. The manufacturers of appliances, furniture, and architectural elements could provide the library with all of the components that the BIM-like solutions are made of: for example, each of the things that go in the bathroom. Any number of manufacturers would be able to provide components for the library, in addition to assemblies of components. In fact, a manufacturer could provide a complete room assembly along with all of the necessary parts and this would be immediately searchable when added to a floor plan.

For component solutions, the pre-designed floor plans stored within the solution library are intended to function as searchable starting point home configurations that the consumer can further refine. The starting point designs themselves would also be provided by developers and home builders, in addition to architects. The designs would simply be a connected group of assemblies and components; so companies can, for example, share assemblies, which are then incorporated into hybrid design solutions that use objects provided by numerous different manufacturers. In this scheme, one can imagine that the developers might provide a base set of highly configurable plans that the consumer then searches through, where architects might create more stylized prototypical spaces that offer solutions of different types.

For all types of representations, the content providers would either use data management interfaces or work with professional integrators to get their designs loaded into the system. In this context, integrators are IT professionals that are familiar with the system and can help companies put their information online, and

can further help these companies transition into newer technologies or production processes. The technical and business processes for loading data into the system are not addressed by this prototype, but the complexity of these processes would, of course, vary depending on the amount of data the provider wants to put into the system and the format in which that data is stored. ²

Each of the various players may potentially have other roles in the system, or take on multiple roles. For example architects might actually collaborate with consumers to generate search requests in an interactive, multi-user environment. And consumers themselves may become the providers of search artifacts that help guide the design of new, searchable configurations, simply by saving their queries. The following chapter offers speculation about the various roles different people could play, and the potential impact of these roles within the design system.

6.3 The Query Interfaces

Collectively, the query interfaces will be straightforward and lightweight to implement. The guidelines developed for this study assume that the interfaces would be developed in Perl, but any web-enabled scripting or programming technology would suffice. ³ Perl is an all-purpose scripting language that performs well and also supports OOP paradigms, in addition to being a widespread web development technology. (Welsh) The Perl scripts attach to a MySQL database, which in turn provides data for the various types of tags that are available to each of the queries. This database also houses information for accessing the resources from the various content providers.

Each of the query interfaces will generate the same kind of output; this means that query output can be shared between the interfaces. This is important because it enables the system to support sessions where the consumer utilizes more than one of the query interfaces to construct the query output. The user evaluations illuminated the fact that people found the interfaces to be useful for

Figure 6.3 (Right) Links Between Interfaces

Users can make multiple pathways through the various interfaces to specify their query

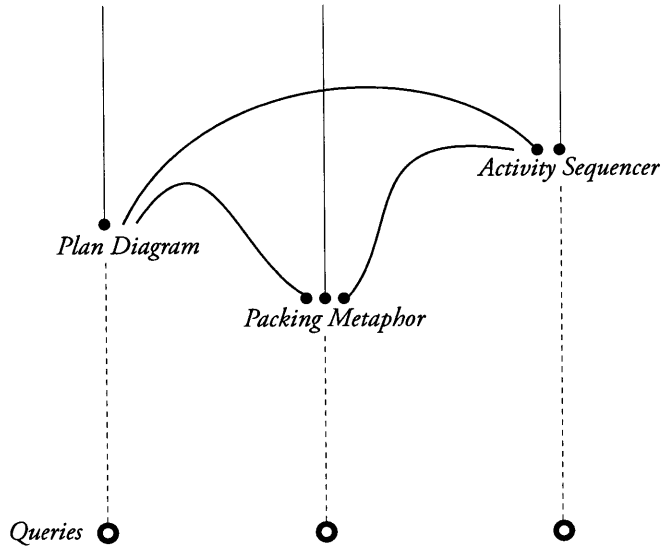


Figure 6.4 (Below) Standard Query Format

The system uses this XML-based RDF structure to pass query information between interfaces and to the search algorithm:

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Config ConfigID="2" Score="10">
  </Config>
  <Config ConfigID="7" Score="28">
  </Config>
  <Config ConfigID="8" Score="30">
  </Config>
  <Description room="Bedroom" id="b01">
    <activity>
      Waking Up
    </activity>
    <multi>
      Kitchen
    </multi>
    <attribute>
      Small But Comfortable
    </attribute>
    <attribute>
      Private
    </attribute>
    <item>
      Bed
    </item>
  </Description>
  <Description room="Kitchen" id="k01">
    <activity>
      Coffee
    </activity>
    <activity>
      Eating
    </activity>
    <multi>
      Bedroom
    </multi>
    <doorway>
      Bathroom
    </doorway>
    <attribute>
      Open Layout
    </attribute>
    <attribute>
      Lots of Counterspace
    </attribute>
    <item>
      Oven
    </item>
    <item>
      Coffee Machine
    </item>
    <item>
      Pots and Pans
    </item>
    <item>
      Kitchen Pantry
    </item>
    <item>
      Shelves
    </item>
  </Description>
</RDF>
```

different types of things, and additionally that they tended to think of the output from the various exercises collectively, as a whole. For example, one user commented that it would be “good to go through the [packing metaphor] for each room in the [diagram]”, suggesting one way that the exercises might be related. Based upon the evaluations, Figure 6.3 shows the links between exercises are proposed.⁴

The output itself, the search query, is structured as a Resource Description Framework (RDF) XML document. (Figure 6.4) RDF is “a standard for Web metadata developed by the W3C... suitable for describing any Web resources, and as such provides interoperability among applications that exchange machine-understandable information on the Web.” (Fensel, 9) Essentially, RDF describes the semantic relation between concepts by means of a simple resource-predicate-object syntactic structure, usually called an “arc”. (Figure 6.5) An example arc in the context of our search tools might be: “Kitchen-Doorway-Living”, which describes a doorway connection (predicate) between the resource (the Kitchen) and the object (the Living room). However, the resource described in an RDF arc can be any sort of concept. In most applications, the resource would be a Web location or URI, because RDF is generally intended to support more intuitive Web processing for the W3C’s current (as of 2007) Semantic Web initiative. (W3C) An example of an RDF arc for a Web location would be something like “http://www.google.com/function-search tool” which describes that the Google resource

functions as a Search Engine. The general purpose of RDF is to make Web resources more intuitively searchable for humans; and in fact, several RDF search tools are already being developed that can query RDF directly. ⁵ This means that the queries, stored as RDF documents, become searchable artifacts themselves. As we will see, there is potentially significant utility to preserving the queries, to get a sense of what the system is being used by consumers to create.

RDF is generally described within an XML document, simply due to the widespread standardization of the XML document structure; however, RDF is not inherently an XML standard. This system utilizes XML to structure RDF queries as they are transmitted between querying interfaces and the search tools, but within the various interfaces and tools the RDF data is structured relationally in the MySQL database.

Our RDF data structure for query output describes a connected, labeled semantic graph where nodes in the graph are room names and rooms are connected together either by doorways, large openings, or indirectly through multiple spaces. (Figure 6.6) Within the room node, any number of attributes and items can be included, in addition to what sorts of activities are expected to happen in the room. This fundamental RDF structure is utilized by each of the conceptual interfaces: the Activity Sequencer, the Packing Metaphor, and the Diagramming tasks.

Due to the inherent scalability problems of the Checklist Specification exercise and because of its relatively poor performance in terms of ease of use and entertainment value, it is not recommended that the Checklists be implemented into the system prototype. In addition, based upon findings from the user studies, a few revisions were made to the query interfaces.

6.3.1 Revisions to the Activity Sequencer Interface

The activity sequencer itself does not require significant revisions: it was clear, approachable, and did not cause any users to make mistakes. However, more attribute tags should be made available, though this study didn't come to any conclusions about what those additional tags ought to be. For most users, as summarized in

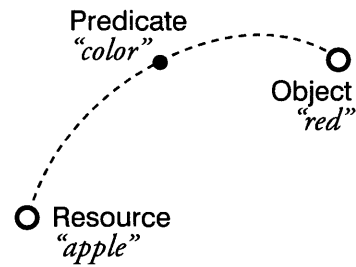


Figure 6.5 RDF Arc Structure
RDF organizes information into arcs, which are also called triples

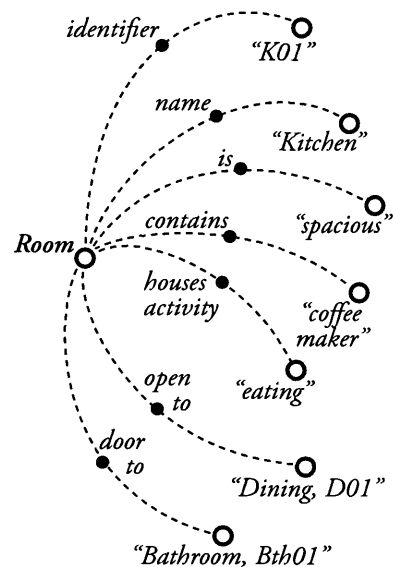


Figure 6.6 Arcs Used by This System
An example of the various attributes that define a room, in RDF

Chapter 5, the attributes that have been provided were adequate. In addition to the attributes, the *transitions between activities* area needs to be more prominent to help signal to users that they can revise those transitions to better describe their needs.

6.3.2 Revisions to the Role-Playing Metaphor Interface

To better support the Packing exercise, the RDF specification for the queries was expanded to differentiate the items listed in the initial *things to be packed* portion of the exercise from any items that may have been added to the subsequent *desired room attributes* portion. It wasn't expected that users would want to put items in the attribute list, but it makes sense for them to do so; thus being able to differentiate the two classes of items is important. Also, the separation of the *things to be packed* items allows this interface to better interoperate with the others, in terms of the underlying RDF.

6.3.3 Revisions made to the Floor Plan Diagram Interface

The Diagram exercise illuminated the need to add unique room identifiers to the underlying RDF structure, to allow the system to differentiate between multiple instances of the same room type. It's a fairly obvious requirement, since many users created diagrams that had multiple bedrooms and / or bathrooms.

The Web interface is more restrictive than the prototype and therefore many of the improvisations that users made will not be possible; for example, attempting to place a hallway room as a connection between adjacent grid cells would not be permitted. However, enough users attempted to specify larger room sizes by opening up adjacent spaces that a "size" attribute was added to allow for more refined matching where possible. To support this, a size value was added to the RDF structure and the diagramming interface will be designed to allow spaces with opening between them to be considered as one room unless they are separately tagged.

6.3.4 Exclusion of the Checklist Interface

Because it was the least popular and least effective exercise, in addition to the scalability issues highlighted in section 4.2.1, the Text-Based Checklist interface is not recommended for inclusion in the prototype.

6.4 Results Listing

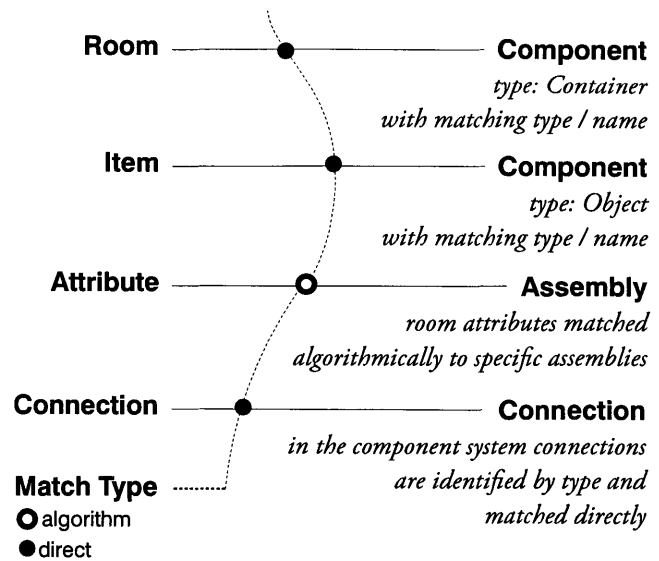
While this study has given particular attention to Component-based representations, the system can search for different types of resources. This means that the query results might have different types of file formats that come up in the results list. As was previously described, there are two basic categories of representations: those that must be indexed and those that are inherently searchable.

Though there is great flexibility in what types of things could be indexed for the search, we assume that the indexed representations would be one of the following types: Images (Jpg, other raster formats), Vector drawings (AutoCAD's DWG format, SVGs ⁶), or a Web resource that uses multiple formats. ⁷ The resources themselves could show different types of content: they could be floor plans, renderings, or photos, or even diagrammatic representations, the system doesn't care. What the system does care about is the auxiliary index that is related to each of the images.

To structure this auxiliary index, the prototype is actually utilizing the same RDF format that is being used for the query. This has a few benefits. First, the search algorithm becomes a bit more straightforward: to search out indexed representations the system is comparing identically structured RDF documents, and simply looking for the best match. Also, the logic for determining the best match would be derived from the search logic described for Component systems in the next section. It also means that the querying interfaces might potentially be retooled into interfaces that allow the content providers to index their design solutions manually.⁸ Still, the processes to support the generation of indexes manually or computationally are part of the content loading process and not described herein; rather, this proposal suggests that searchable representations be used to circumvent the problem of auxiliary indexing altogether.

Figure 6.7 Mapping Queries to Component Based Representations

The search algorithm in the proposed system utilizes the relations shown here to map queries into more literal descriptions



Object-oriented Component-based representations are directly searchable. In this case the searching process, as described in the next section, utilizes a mapping program to parse an incoming RDF query and search for the best matching configurations housed by the Component system. Figure 6.7 shows how the system maps RDF to Component-Based data.

6.4.1 Component System Structure

For the search procedure that was prototyped, the Component system was structured into a relational database format in MySQL, which defined the components and their assemblages. (Appendix 2, A2.1) Twenty Five (25) basic design solutions were loaded into the system; these solutions were designed for the same multi-family housing context that the user studies were based upon.⁹ Components have dimensions, type designations, and other manufacturer-provided metadata. The assemblies contain a hierarchical listing of components with specific locations that is related through labeled connection types. Each type of component has at least one interface upon which connections can be made; these interfaces describe type, range, and fit requirements as described in Chapter 3. (Figure 3.6)

One complicating factor within the prototype is how to transition the consumer from the searching interface into the Component system interface. Our prototype, being developed within a fast and

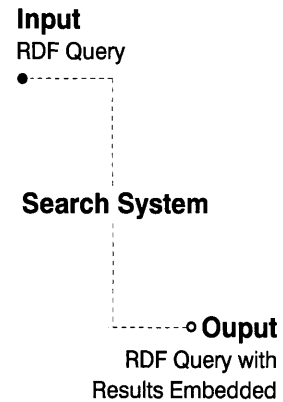
accessible relational database, is meant to support the development of a web-based component design system which roughly follows the structure of the exercise detailed in Chapter 4, and which the user could simply click into. If the Component representation was something like Autodesk's Revit BIM, the BIM document might need to be downloaded or delivered through some other type of interface – this is a more complex delivery problem which we're not addressing.

6.5 The Search Algorithm for Component Systems

The prototypical search tool described in this section was developed in C#. ¹⁰ The input into the tool is an RDF query as described previously. As shown in Figure 6.8, the output is the same RDF query with search results and relevance information embedded directly within.

To find results, the search tool interfaces with a relational, MySQL-based component system. ¹¹ In truth, the MySQL implementation represents an incomplete component system, because while the core data structure is fully implemented, the system doesn't have a completely functional prototype of the actual modeling or drawing environment that would be built on top of it. So the prototype is simplified and, therefore, it may not be possible to generalize from the results achieved to other types of component systems, like BIM, that store their data structure in proprietary, binary documents. However, the object-oriented specification of BIM and other component systems correlates with the database structure that was developed, so rather than spend time trying to figuring out the details of interfacing with any one specific solution, we've considered our simplified and open-source database to be representative of the general structure of component-based systems, with the goal being that our findings would be thus able to be generalized.

Figure 6.8 Search System Output
The search posts relevancy information directly within the query itself



6.5.1 Searching and Sorting

Both the Component system and the RDF queries are complex, structured data-sets. As such, the search algorithm is a high-level search, a relational search that utilizes identifiers to group concepts and find properties.¹² This is the same paradigm upon which SQL search algorithms and thus SQL languages are built. In this case, and because a SQL technology was used to organize the searchable data, the search actually utilizes SQL queries to access the data; thus the search is both SQL-like, and SQL-based. By incorporating the ability to search through a discrete list of content providers, the search process described herein functions as a Federated search, where multiple, disparate data sources are searched and then merged together.

Relational, structured databases are built out of a collection of tables; these tables are like a flat listing and are not inherently hierarchical like the RDF queries and component system's specification require. Within SQL, the hierarchy is established through the specific types of relationships drawn between the tables, where for example one table called "component" might be a child to another table called "assembly". Additionally, a simple hierarchical structure can be created within a single SQL table through the use of self-referencing structures, a well-known methodology which was utilized in the SQL design to allow for the specification of an increasingly granular type hierarchy.¹ (Appendix 2, A2.1) Taken together, these references allow the system to describe the hierarchical structure of the components with little difficulty.

Within the search application itself, the RDF data is organized into internal class structures that include basic data sets for the various child elements and properties; for example, the objects a room contains. In both the MySQL specification and the internal data classes, the search algorithm takes advantage of current technologies that utilize well-known methods for structuring and storing data for easy access. In both cases, data that is stored on the hard disk is organized into either Binary Search Trees (BSTs, Figure 6.9) or sorted lists upon which extremely fast binary search operations (Figure 6.10) may be run.¹⁴

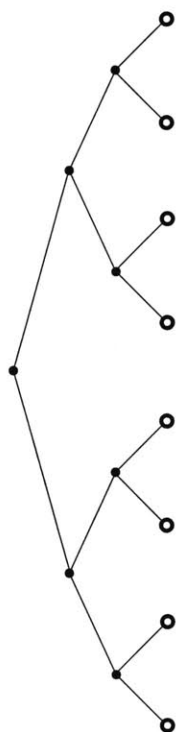


Figure 6.9 Binary Search Tree

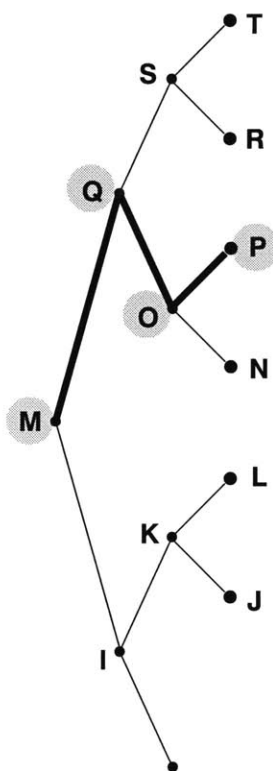


Figure 6.10 Binary Search Algorithm
Here, the algorithm is looking for the letter "P"

To enable the search algorithm to find partial matches to the user's query, the query is broken up into individually searchable parts. Partial matches are essential for making the search work because they enable results to come back to the user when searches are not completely matched. Allowing the user to learn what criteria are not available from reviewing partial matches is one of the most fundamentally educational aspects of the search cycle.

Following the structure of the RDF query, the searching is broken up into the following stages: complete room graph, room pairs, single rooms, and individually, each of the included activities, items, and attributes. Each stage of the search, if successful, contributes points to an overall score for the solution being queried. In application, the complete room graph is searched for at the same time that the room pairs are: if all pairs are found the room graph is considered to be a complete and the solution is given an additional score multiplier. Individual rooms are searched one-by-one and have a relatively small impact on the final score. Within each of the rooms, all of the properties (items, attributes, and activities) are searched for individually; each type of property is given a different weight. The item search is quite straightforward: the RDF item label is compared to the component name and its complete, self-referencing type designation.¹⁵

The attributes are a little more complex: for each specific attribute, the overall room assembly is analyzed according to the semantic definition of that attribute.¹⁶ Processing the attributes

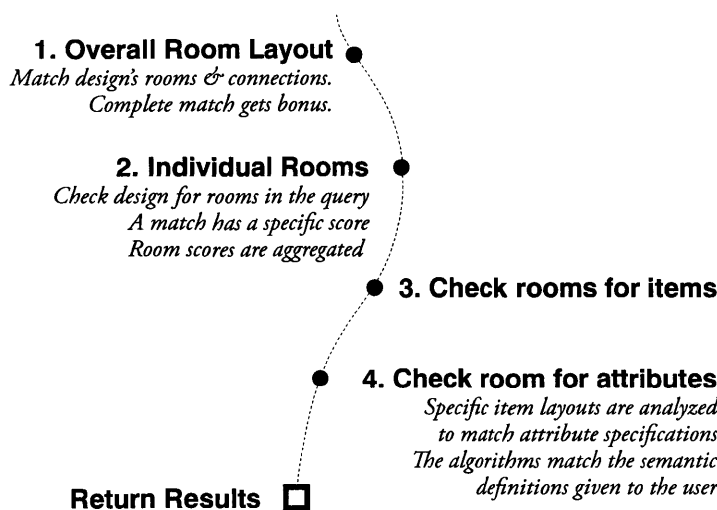


Figure 6.11 Search Algorithm Stages
 The query is searched in sections, to allow for partial matches

is problematic in a few ways. In the prototype, the attributes are each programmed individually, which presents a scalability problem because, as semantic definitions are added to the system, the search code would need to be updated to support them. Also some attributes are much harder to characterize quantitatively than others are. The tag “Efficient” for example could mean many different things - it is hard to think of a brief and general description. As a rule, the definitions that the system provides are rooted in the facts of physical objects and relationships. So, to the prototype, efficient means “Surface/sitting objects connected or close to each other.” But future implementations may be better served by making the attribution process more algorithmic, or perhaps room-specific, where room components could be compared to ideal-case layouts as a more empirical measure of true efficiency.¹⁷ Still, the total number of semantic definitions required may not be that large, and as the user studies showed, only a few users felt that the base definitions were inadequate. This is something that another round of evaluations, done with web-based prototypes could help tease out.

The different search stages are given different weights and thus influence the score differently. In the current system, the weights are adjustable. Based upon evaluations, the weights shown in Figure 6.12 were used for the search algorithm. Using weights as shown leads to a phenomenon where weaker queries, or queries with less criteria and therefore less specificity, return on average lower relevancy scores than stronger queries. This scoring methodology is a bit different from common Web search tools that report relevancy as a percentage of the search term, but it makes sense in terms of the design search, because it offers a reflection of the specificity of the query. If the user has only specified a few criteria, the search is not going to arrive at meaningful results, so a lower potential scale for weaker searches allows the system to recognize weaker searches by looking at the score alone. The system might then engage the user in tutorials, or point him to a different conceptual interface. Additionally, one of the nice things about each of the conceptual interfaces is that they present exercises that have a clear beginning and ending. Each of the volunteers had a clear understanding of when the exercise was “done”, and the vast majority exhibited a desire to complete the exercises.¹⁸ In terms of

Room Layout, Complete	12
Room-to-Room Connection	6
Room	1
Room Attribute	9
Room Item	6

Figure 6.12 Weights for Search Criteria
 These weights are adjustable to allow for further refinement. The weights shown here are those used for the system analysis

the queries, completed exercises mean stronger searches.

6.5.2 Reporting Results

Matching resources are reported directly into the RDF query structure, in two modes. The first and default mode is the summary results listing which simply provides a ranked and ordered listing of design solutions. This is how the search tool replies when a new query is posted. But the tool allows for more detailed relevancy reporting: if the user selected any one of the design solutions, the search tool sends another RDF reply that omits the summary listing but includes a detailed breakdown of the scoring for the query. These more detailed replies allow relevancy and scoring information to be shown directly within the conceptual querying interfaces, because the search algorithm states exactly what parts matched and how significant each of the matches were. This is inherent to the search cycle and an important facilitator for design guidance because the user can, for example, see quickly what parts of their query are not getting any matches and therefore might need revisiting. In fact, posting the relevancy visually, into the query composition, functions the way that an adaptable and well-specified embedded critic ought to, without requiring any additional, complex programming.¹⁹

In addition to the problems associated with making sense of

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" dc="http://purl.org/dc/elements/1.1/">
  <Config ConfigID="4" Score="43">
  </Config>
  <Description room="Bedroom" score="2">
    <activity>Check Weather</activity>
    <activity>Dressing</activity>
    <multi>Kitchen</multi>
    <doorway>Bathroom</doorway>
    <attribute score="5">Private</attribute>
    <attribute score="5">Spacious</attribute>
    <attribute score="0">Abundant Natural Light</attribute>
    <item score="6">Shelves</item>
    <item score="6">Closet</item>
  </Description>
  <Description room="Kitchen" score="4">
    <multi>Bedroom</multi>
    <attribute>Lots of Counterspace</attribute>
    <attribute score="5">Spacious</attribute>
    <attribute score="0">Abundant Natural Light</attribute>
    <item score="4">Island</item>
  </Description>
</RDF>
```

Figure 6.13 Relevancy Scoring
This is an example of the system's output
An RDF-based query has specific relevancy
scores for a specific solution embedded.
Both scores and solution identifier are
highlighted

attribute tags, there are potential scalability issues with the posting of summarized results listings into the RDF structure: if the system were to get large with hundreds or thousands of potential matches being reported, the text-heavy RDF format would not really be an efficient way to pass the information around. In this case, the search algorithm, as prototyped, could simply stream a pre-formatted results list directly to the query interfaces.

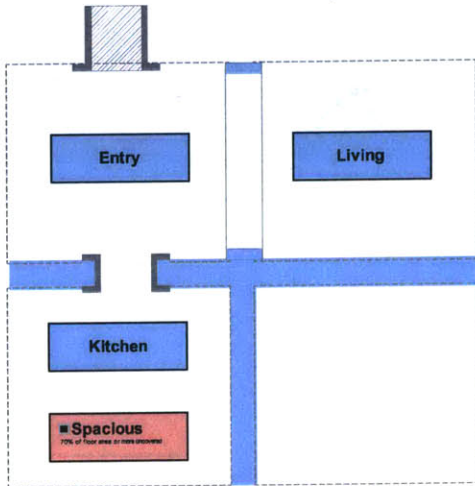
6.7 Visual Explanation of the Search

The following pages provide a visual series that shows how the search works. The example is a simple query, built by the consumer using only the diagramming interface.

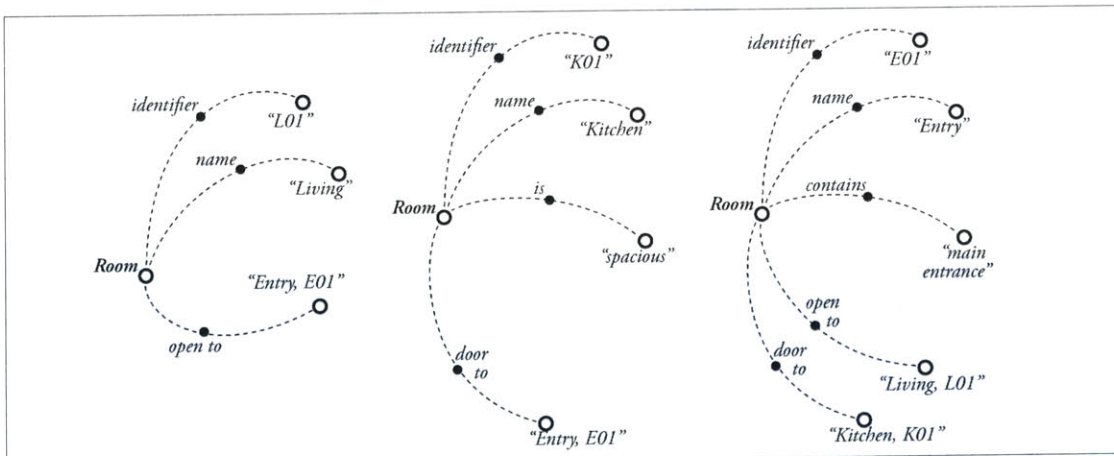
How The Search Works



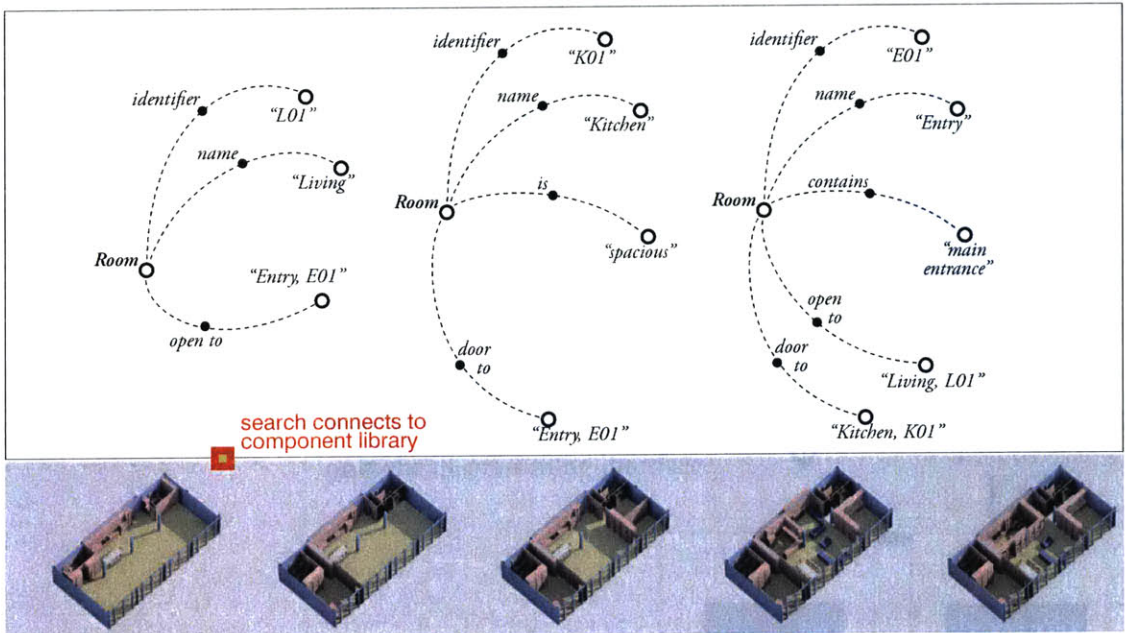
user builds query



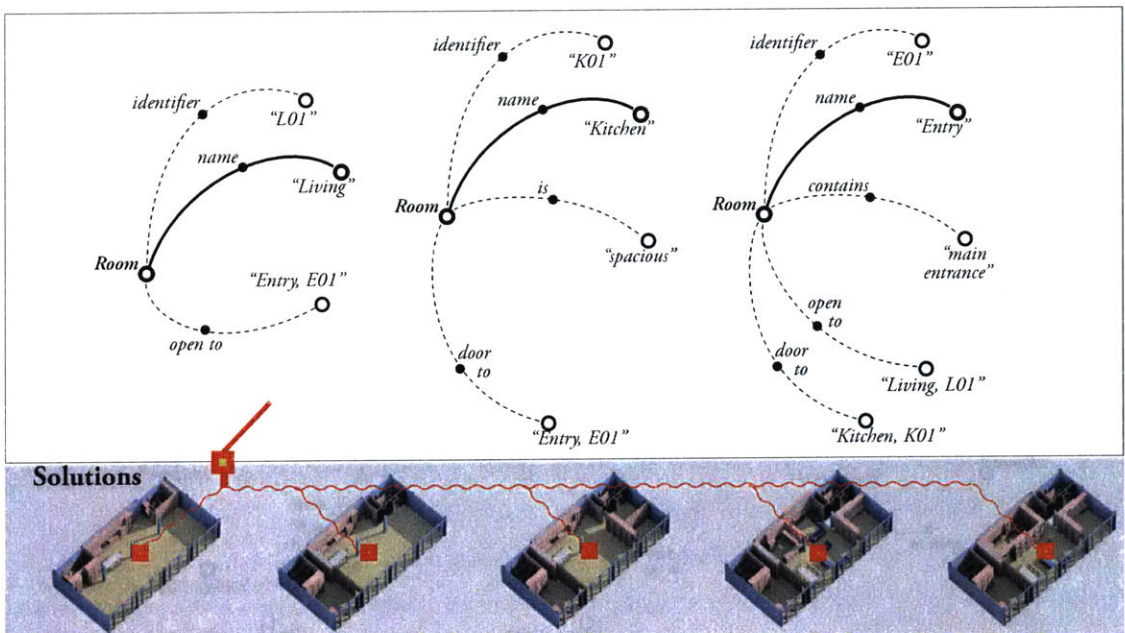
user submits query



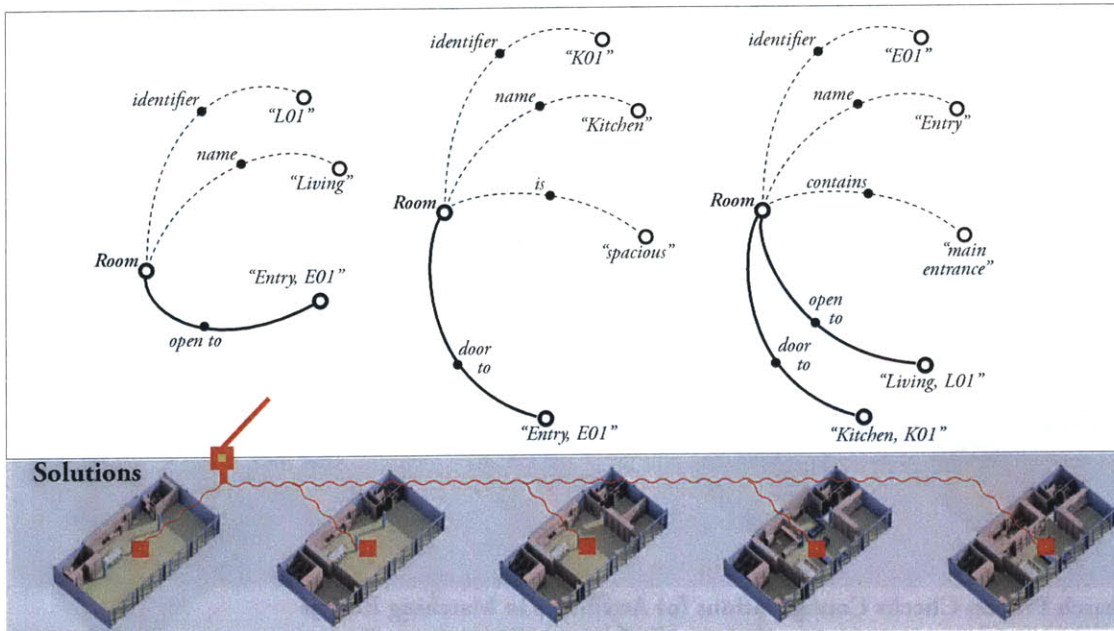
Search System Reads RDF Format



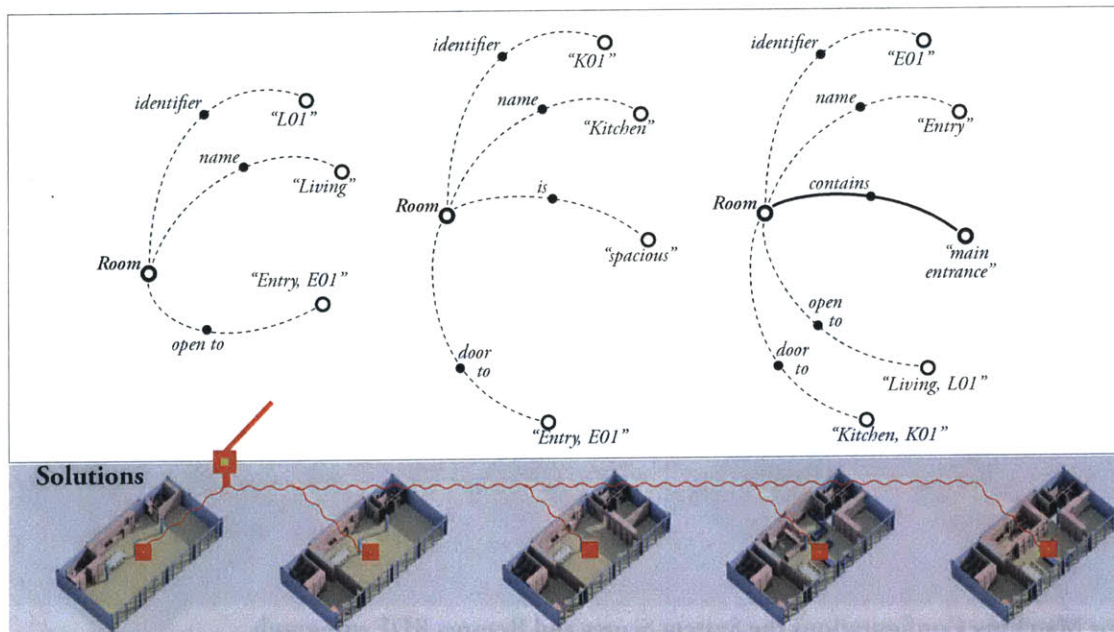
Search System Connects to Component Database



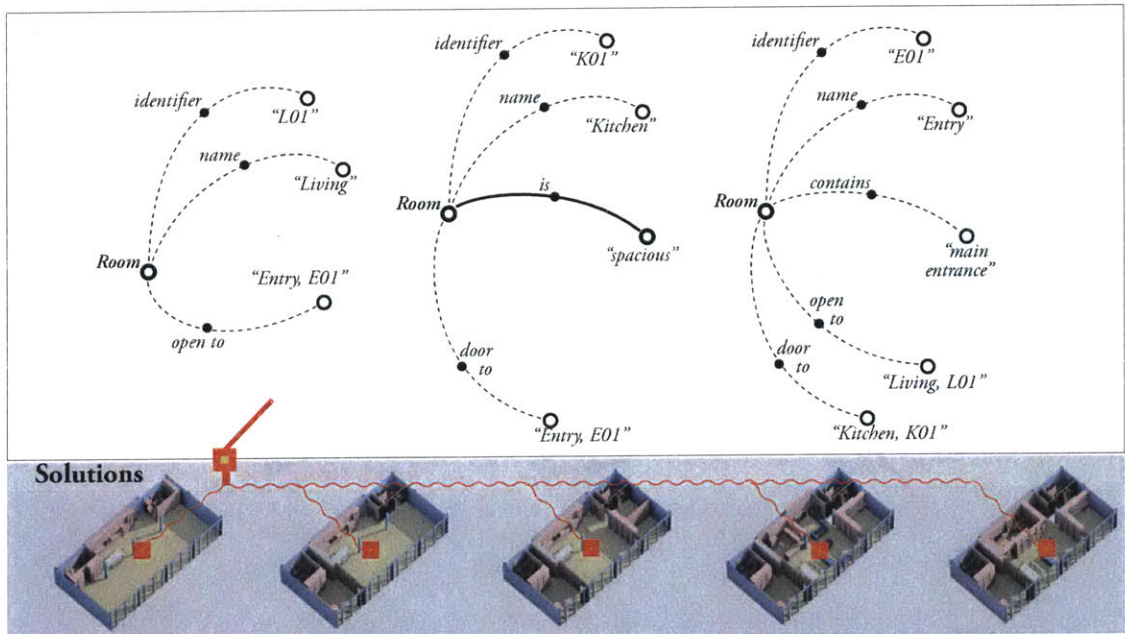
Search Process Checks Configurations for Matching Rooms



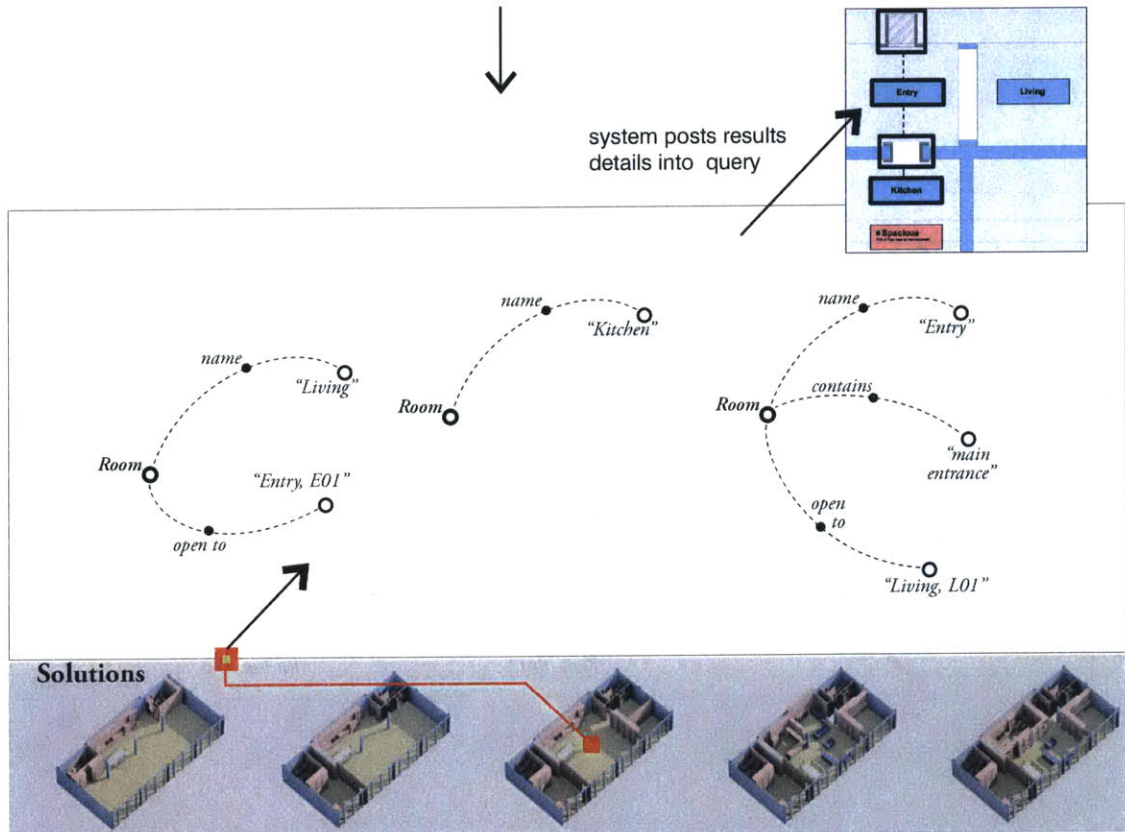
Search Process Checks Configurations for Room Connections



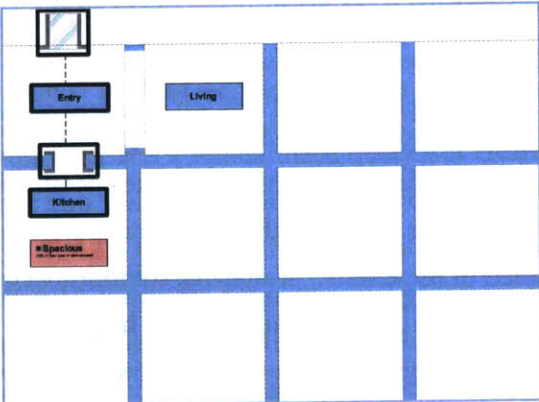
Search Process Checks Configurations for Objects in Matching Rooms



Solutions
Search Process Checks Configurations for Attributes in Matching Rooms




Solutions
For Matching Configurations the System Scores and Returns RDF sub-graph





Search Again Add Activities Pack Your Stuff


Other Results

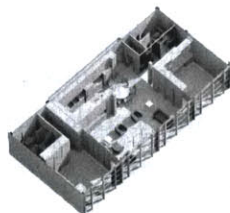
- [2 BR / 2 Story Country Home by Toll Brothers](#)


- [Two Bedroom with Study Area - HomeSeeker.com](#)


- [Three Bedroom w/ Large Garage and Boat Storage - VacasHomes.net](#)


- [Two Bedroom, Modern - With Labels by Dierdra Homes](#)

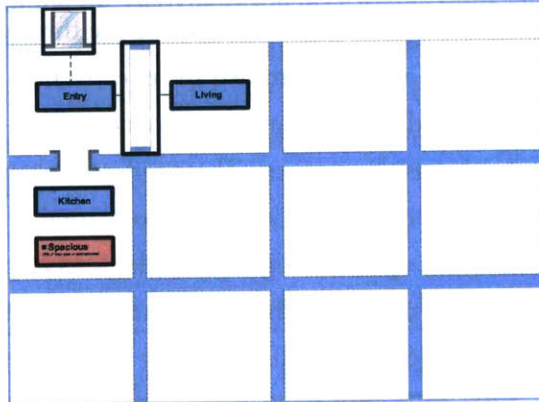




Basic Two Bedroom Unit
by MIT house_n
All rooms found. Kitchen connects to entry through doorway.

Query Interface Shows the Specific Relevancy Visually


Matching Configurations are sorted by score into the results listing
Best Matching Configuration is shown in more detail





Search Again Add Activities Pack Your Stuff


Other Results


- [2 BR / 2 Story Country Home by Toll Brothers](#)


- [Two Bedroom with Study Area - HomeSeeker.com](#)


- [Three Bedroom w/ Large Garage and Boat Storage - VacasHomes.net](#)


- [Two Bedroom, Modern - With Labels by Dierdra Homes](#)





Basic LOFT Unit
by MIT house_n
All rooms found. Living Space Open to Entry.

User Can Click on Other Results to See How They Match Differently

6.6 Evaluating the Effectiveness of the Search

Two sets of tests were run to determine how well the conceptual queries map to the literal solutions and what level of granularity the algorithm provides. The findings detailed here are encouraging but it's very important to note that both the query and search space were highly constrained in these tests. Searchable solutions were limited in scope to one specific design context. This helped to make more straightforward the fundamental comparisons in this research, but it also means that each of the searchable resources has many attributes in common. The search space was thus constrained. A more complete and rigorous analysis would be requisite for getting a true sense of how well this search actually performs in a larger implementation. Constrained tests like those detailed here are useful for determining basic functional effectiveness but they do not adequately represent the variance and therefore the true differentiability of a search space on the order of that in the proposed system.

The first set of tests was designed to give a sense of how well the component system mapped to the conceptual queries in the context of a specific room. These tests used three different source queries which were representative of the output from each of the three query interfaces. The second series of tests examined the effectiveness of the search with RDF-encoded queries that were generated from a selection of the actual volunteer's exercises. Three different examples were encoded for each of the conceptual interfaces for a total of nine. These searches were then run against a slightly broader search space to get a sense of the granularity of the search by comparing the best and worst matches. The motivation was to discover how well the search criteria were able to differentiate solutions in the search space.

6.6.1 Search Analysis Series 1: Mapping

To see how well the search prototype was mapping the conceptual to the component based representations, a constrained search space was utilized: 10 design configurations were loaded into the component system. These configurations were identical with the

exception of the kitchen, for which a range of solutions was made available. Thus, the only variance in the search scores for the configurations was in the specific relevancy of each kitchen. For each conceptual interface, a standardized query was created, manually, to test out how well the search tool mapped to components (see Appendix 2, A2.3 for the RDF of these queries).

Table 6.14 shows the range of scores generated by each of the conceptual exercises for the 10 different kitchen layouts. The Activity Sequencer scores are higher because that exercise’s queries specify additional searchable information about the other rooms in the search space. Likewise, the diagrams generated lower overall scores when queried independently, because only the room connections are specified within them, not attributes or items. Overall each of the exercises has a fairly small deviation which was expected from running such a constrained sample: 2.95 for the Activity Sequencer, 4.08 for the Packing Metaphor, and 3.44 for the Diagram. In terms of a search operation, these numbers represent a field where a few items are good matches, and a few items are weaker matches, with the rest being fairly average. But there are a few other, very interesting things to discover in this data.

Examining the Activity Sequencer results more closely, we see a compelling example of the way that this search system can provide design criticism and guidance inherently through the relevance numbers. In this query we see from the mapping details that the user is searching for a kitchen that is spacious, has an island, and has a

Score Rank	1	2	3	4	5	6	7	8	9	10
Activity Sequencer	63 <i>table</i>	63 <i>table</i>	66 <i>spacious</i>	66 <i>spacious</i>	66 <i>spacious</i>	66 <i>spacious</i>	66 <i>spacious</i>	66 <i>spacious</i>	72 <i>spacious island</i>	72 <i>spacious island</i>
Packing Metaphor	22 <i>spacious oven fridge</i>	22 <i>spacious oven fridge</i>	22 <i>spacious oven fridge</i>	22 <i>spacious oven fridge</i>	28 <i>spacious oven table fridge</i>	28 <i>counter space oven table fridge</i>	28 <i>counter space oven table fridge</i>	28 <i>spacious oven island fridge</i>	31 <i>spacious private oven fridge</i>	34 <i>spacious island shelves oven fridge</i>
Floor Plan Diagram	4 -	10 <i>door to living</i>	10 <i>door to living</i>	10 <i>door to living</i>	10 <i>door to living</i>	10 <i>door to living</i>	10 <i>door to living</i>	10 <i>door to living</i>	16 <i>door to living island</i>	16 <i>door to living island</i>

Table 6.14 Analysis Series 1: Mapping
10 Different Kitchen layouts were searched using manually-coded RDF queries

table. The system returns three different relevance scores, where 63 = table, 66 = spacious, and 72 = spacious & island. Simply by analyzing the designs during the search process, the system has shown clearly that you can't have a spacious kitchen if you also fill it with a table! However, you *can* have an island and a kitchen that is still spacious. Because they are the more complete match, results that offer both spaciousness and an island come back with the highest scores. Thus, the system has effectively utilized the expertise captured within the design of the component assemblies themselves to determine what tradeoffs the user needs to make. This type of functionality would have been far more difficult if not impossible to try and make a machine smart enough to reason through. By using the search space as a criticism source, the system lets computers do what they are good at doing (search comparisons), and lets the designers who create the search space provide the expert criticism simply by designing livable, functional spaces.

The Packing Metaphor example, on the other hand, highlights some challenges. Here we see, in the solutions that scored 28, four different types of matches that came back with the same relevancy ranking. This seems to indicate that the search didn't make enough separation between solutions that were not equivalent, and may not have been of equal importance to the user. One way to address this issue would be to allow the user to specify, in the query, if something is more important. The system could then use this importance score to further differentiate the results.

The Diagram query functioned as expected. The lowest scoring result didn't match the kitchen space at all, while the highest scoring results both connected the kitchen correctly to the living space, as was specified in the diagram, and also matched the island within the room.

6.6.2 Search Analysis Series 2: Granularity

The second series of tests (results in Figure 6.15, Page 120) were intended to get a sense of the overall granularity of the search algorithm, by using the queries that users designed during the interface evaluations. Apart from being a nice way to tie the user

and search analyses together, it allows us to test out the search with the types of queries that actual users make. For each exercise, three user compositions were selected and encoded into RDF queries; these queries were then used to perform searches (Appendix 2, A2.4) for queries). The search space for this series of tests was different from the initial test: here 10 different design solutions were loaded that have the same basic layout but varied room selections for all types of rooms. For each query, the best (highest) and worst (lowest) scoring results were listed. Figure 6.15 summarizes the results.

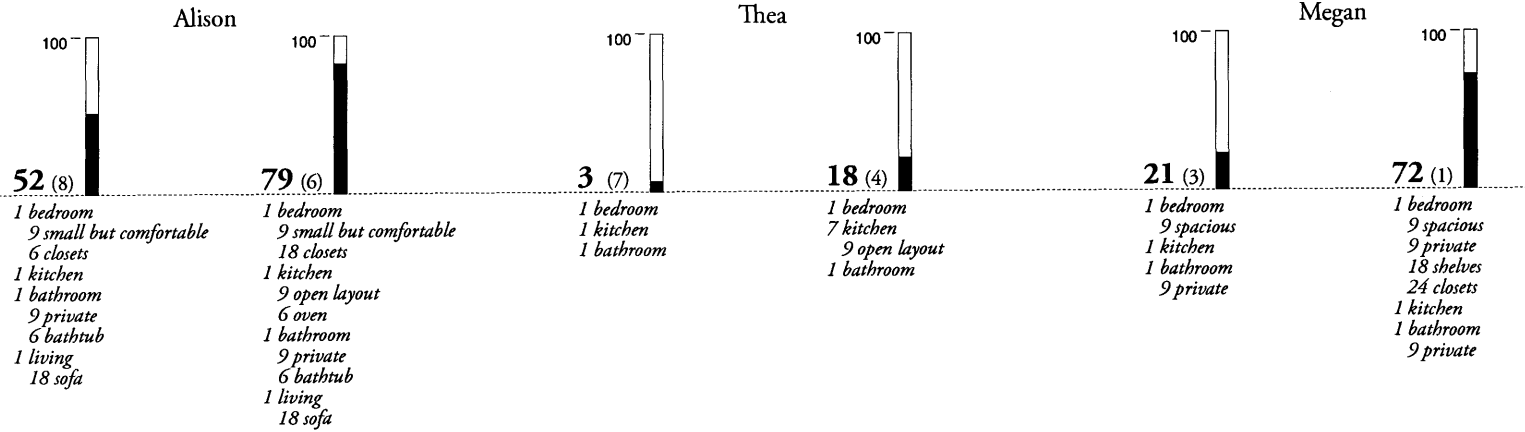
Again the Activity Sequencer performs well, because it encourages the detailed specification of multiple rooms where the Packing Metaphor focuses on one rooms, and the Diagrams the users made only dealt with room connections. But a quick examination of the data further suggests that the effectiveness of each of the interfaces varies with the user. Still, with the exception of the Packing Metaphor, all of the queries generated a reasonable best match and a proportionately typical worst match. In other words, even in cases where the best matches were weak matches, they were still highly differentiated from the worst matches, with few exceptions.

The notable exception was Jeffrey's Packing Metaphor query, which scored even 10s across the board. This was because his only searchable criteria were: "abundant natural light", "spacious", and "closed layout". The solutions for this evaluation didn't have window components included, so as it turns out, natural light wasn't searchable (See Appendix 2 for system details). And while all of the bedrooms were evaluated to be spacious; none were designed with a closed layout. The other two Packing Metaphor queries generated a highly bipolar result set where the best scores were 16 and 19 and the worst score was 1. Taken together, these three queries suggest that the Packing Metaphor is a bit weak on its own, and as was observed by a few of the users themselves, would function better when coupled with the other interfaces.

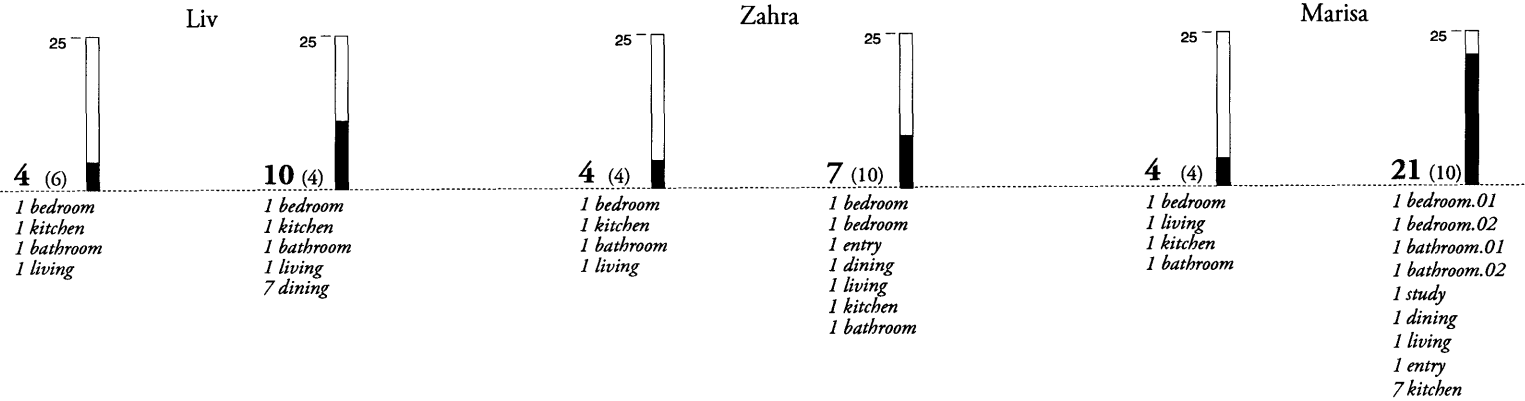
The Floor Plan Diagram queries generated a moderate variance and functioned as expected, but did suffer a bit with the highly constrained search space. This is because the layout of the user's diagrams was not constrained, but within the evaluated search space, the layout of rooms did not vary; only the connections between the

Key: Scale
Score (Solution #)
score breakdown, line by line

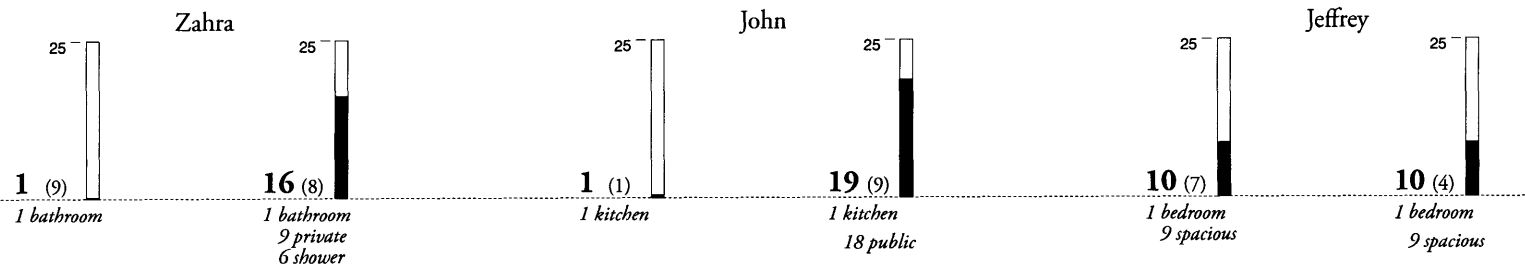
Activity Sequencer



Floor Plan Diagram



Packing Metaphor



Solution's Score:

Lowest

Highest

Lowest

Highest

Lowest

Highest

rooms varied. The results show this clearly, where the best scores are those where the system was able to match a part of the diagram's organization. This doesn't reflect negatively on the potential of this highly popular interface; however, it does confirm that more detailed testing of the search algorithm is required.

(Facing Page)
Figure 6.15 Analysis Series 2: Granularity
User queries were encoded to search against a constrained solution space

6.6.3 Search Analysis Summary

With the revisions that were made subsequent to the user evaluations, this basic analysis of the search algorithm led to positive results. The search algorithm demonstrated the ability to map from the conceptual to the literal representations effectively. In addition, the results that the search tool generated were well-differentiated. Even more promising was the ability of the searches' relevancy information to communicate, to the user, the tradeoffs inherent in the search space. This is a positive result that confirms the thesis of this work: that a search tool can support design discovery. With the addition of tuning mechanisms to allow users to further describe the importance of specific criteria, these initial tests were positive enough that further, more elaborate analysis is a worthwhile venture. A simple web-based implementation of the interfaces that additionally supports the links between the various conceptual exercises, as described in Figure 6.4, would be a great place to start.

Again, these evaluations were simplified tests of the system using a highly constrained solution space that didn't have a tremendous amount of variety. This was useful in that it allowed the analysis to highlight and evaluate the more fundamental aspects of the search process. But the findings detailed here beg for a deeper inquiry; it would be fascinating to see how well the diagramming exercise functions in a solution space that includes different building layouts and different building contexts. From what was learned, it goes without saying that more complete testing is necessary, but also worth doing.

1. For more specific technical information regarding the various components of the search system, refer to Appendix 2.
2. The problem of helping developers, builders, and manufacturers to load the library is significant. To actually happen, it would require the buy-in of each of these players into a new service channel, so the problem is not necessarily a technical one as much as a business one. Manufacturers would have a natural incentive to gain access to the untapped market demand for customizable homes. In theory, developers would like to gain access to that same market as well. In application, the process of companies incorporating solutions into a centralized system of this type would probably be an incremental one, where integrators help companies to slowly add more and more resources into the system as both the system and their own production technologies mature.
3. Perl was selected because it is flexible, robust, and also because it's the author's preference. In truth, the conceptual interfaces would be quite simple to program in other platforms like PHP or ASP.NET, or Flash.
4. Establishing links between the conceptual interfaces would help the user to see how the information is related, but people could still go through the exercises selectively, and of their own accord.
5. An example of an RDF query tool is SPARQL. (W3C 2007) It's designed for straightforward textual searches and is not as robust as fully a relational SQL implementation, so it isn't of utility to our search algorithms. SPARQL design is constrained to RDF arcs where our data is more complex, but it could possibly be useful in making comparisons between RDF queries and indexes in our case.
6. From the W3C specification: "Scalable Vector Graphics (SVG) is a language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text." (W3C 1/14/2003). It could be useful for providing web-based visualizations of database-stored Component information. However, while SVG is interactive and can be animated, it does not support 3D information and therefore isn't a great candidate for component visualization. Even so, it's becoming increasingly popular, and can be made to emulate 3D functionality, though not easily.
7. We expect that commonly, Web based resources would be something like a Web page that provides a design concept summary in addition to a textual description, as well as a few floor plan images, and a sample photo. This is similar to the home concept summary pages that many builders currently provide online – the search tools here simply provide a better way to find these summary pages.
8. The assertion that the query interfaces might be adapted for use as indexing tools for content providers is speculative; none of our evaluations tested that out. Still, it seems plausible, and just as the interfaces promote

- better searches by providing exercises with clear goals which encourage a certain amount of interaction, so too could they promote better indexing.
9. By utilizing the same context for the prototype implementation we can use the volunteer's actual compositions to test out the effectiveness of the search tools.
 10. C# was chosen because it's a well-structured and high-performance language; other languages like Java are certainly an option.
 11. The choice of MySQL was made for several reasons. First, the prototypes being developed herein are part of MIT House_n's Open Source Building Alliance and as such, the tools are intended to be open source. So it makes sense that the database be open source as well. MySQL is also increasingly popular and robust choice, not to mention a free one.
 12. SQL searches are high-level because they operate upon highly structured data. But of course, the database access functions have built-in low-level searching functionality to help make the databases more high performance. In the example of a relational database, table records are indexed and stored on disk in pre-sorted data structures; these pre-sorted structures allow for logarithmically fast binary search algorithms which are generally much more efficient than simple, linear searches.
 13. Self-referencing indexes are a well-known and fundamental methodology for allowing relational lists to represent hierarchical data. The methodology is simple: one column in the listing, let's call it "ItemID", is a unique identifier for the record. Another column "ParentID" stores the ItemID of that record's parent record. Thus a hierarchy is established.
 14. Binary search algorithms reduce the search time by splitting the list or tree to be searched in half (hence binary) recursively until the solution is found. This only works on lists that are pre-sorted and in the case of trees, it works more consistently when the tree is balanced. (Sedgewick)
 15. The Component Type specification as detailed in Appendix 2 is self-referencing which enables hierarchies to emerge through a simple parent-child relationship. So a Queen-Sized bed could be stored by the system as Object>Laying Surface>Bed>Queen-Sized. In the context of the search, this helps numerous individual components match up with more general, type-based search terms. Also, it should enable more complex search analysis because types can branch hierarchically as well.
 16. As detailed in Chapter 4, the room attribute tags contained semantic definitions that explained what the qualitative tag meant in quantitative terms. For example, the tag for "Open Layout" contained the following description: "Opening to adjacent spaces without doors." See Figure 4.2 for the tags and definitions that were provided.
 17. Only a certain amount of time was given to the users for each exercise, to keep the sessions from running too long. Still, 10 out of 12 users opted to finish up their designs before moving on to the next task, even though they were advised it would cause the entire session to take a little extra

time.

18. For an example of a design standards methodology in universal kitchen design, see Xiaoyi Ma's Thesis (Ma).
19. I argue that the process of mapping relevancy information back into the source query is actually an ideal scenario for managing design criticism as part of the typical search results evaluation process that all users are familiar with. And showing visually what specific elements of the consumers design were well-matched and which elements were not helps the user to determine exactly where their own specification is not functional or is just off. Additionally, the criticism itself is embedded within the metadata and as such can be described or represented by any number of different interfaces, and the criticism source is the evolving, growing, and expert designed collection of solutions that are being provided. This allows the criticism to adapt and grow with intelligence.

7 *Where Do We Go From Here?*

7.1 *Summary*

This work has demonstrated that the well-known structure of a computational search engine can be overloaded to facilitate designing by users who are not experienced designers in the domain of the search space. The specific context of this research was in the development of a participative architectural design system that enables consumers to make certain design decisions about their home. In this context, the search engine functions as a first stage of designing and allows for different types of needs assessment and conceptualization through a collection of unique query interfaces. The search was evaluated through user studies, which confirmed that the interface's conceptual structures not only made the exercises fun and easy to learn, but effective in framing design needs as well.

This research further investigated the potential of new object-oriented building representations as a more intelligent and computable format for storing design solutions. The analysis explored the ability of these component-based representations to describe physical realities with enough specificity that searches could be conducted effectively and without additional and complex indexing processes. The preliminary evaluations described in this work show that the information in component-based representations does map well to the proposed conceptual query structure and also that the search space, even when highly constrained as was the case in our analysis, yields effectively differentiated results.

Additionally, the retooled search process was demonstrated to have the ability to provide design guidance and criticism inherently through a rich and highly specific relevancy reporting mechanism. This finding suggests that complicated, expert, machine reasoning algorithms are not always necessary in this context, because the search process allows the system to directly describe the intelligence embedded within the searchable solutions and to use that description to provide criticism to the user. Given that the searchable solutions would be created by architects and other designers, this means

that the search system incorporates people into its algorithm, and lets people do something they are good at doing (designing and evaluating layouts), all while letting computers do something *they* are good at doing, namely, searching through structured data.

In a sense, this research is predicated upon the notion that the development of software to support design processes, in any context, ought to be done thoughtfully and with a critical sensibility about the fundamental methodologies and technologies utilized. In the context of non-expert home consumers, the most essential aspect of designing, particularly in the early stages, is the discovery of needs, values and preferences to help make decisions later on. It is difficult to understand why commercially developed tools incorporate only flat, simplified narrative interfaces or highly complex modeling environments to support the early stages of a design process where other, more computationally mature technologies like the search can be utilized for the same functionality with far better results. Rather than rush into the development of tools with the latest and greatest functionality or flashy representations, one should take the time to examine how well the *computational* structure of the proposed solution maps to the *conceptual* structure of the task. This work has made strides to confirm the thesis that computational searching maps well to human designing.

7.2 *The Query as Artifact*

An interesting extension of the work described here would be to explore the different things you can actually do with the queries, other than directly searching with them. Preserving the queries as the residual artifact of a search operation creates many exciting possibilities, and has several immediate uses. First of all, the query could be posted as a design request into an online community of architects or designers who wish to make their services available to users. Consider the following example: a user specifies an elaborate search diagram with several activity sequences also described. But the system only returns a few results, none of which the user finds to be favorable. Rather than leave frustrated, the user posts the query as a design request that gets picked up by a local architect. The architect

then works to develop a solution that meets the user's needs more directly than those that were previously available. This actually opens up a new service channel for architects to directly engage with home consumers, and also helps to keep the solutions in the search space evolving.

Being recorded into permanence, outside of the immediate context of the search cycle, means that the query itself becomes an informative and computable artifact. Why call it an artifact? Because such queries, when preserved and made accessible, are a cultural, personal, and anthropologically meaningful fragment; they are a piece of history. And being structured in RDF, these queries would be easy to search for, not just to search *with*. After some time, collections of queries could become highly informative repositories of design intelligence that track the evolution of space use, or for example, the way design preferences change over time in response to new technologies and thus different lifestyles. When used as a design guide for professional architects or other designers, this sort of repository has immense potential to help keep living spaces efficient, relevant, and well-adapted.

Another potential use of the preserved queries is as an organizational workspace for the users themselves. One could imagine that even for users who aren't trying to buy a home in the near future, having a profile within which their queries are stored might provide a place for them to organize their needs and values over a longer period of time. This resonates with the research that was recently done by Jennifer Beaudin (MIT House_N), which examined the non-expert homebuyer's design goals over different time frames and in different contexts. Saved information could help the user make a home purchase at some later date, or outside the system itself. This point also came up in the user studies, because Thea, who is renovating a house in Connecticut, noted:

“we drive down and look at [the house] and take all these measurements and draw everything out then we go home and we think about it, and look through magazines, and then we have all these great ideas, and then we go back and look at the actual space again and we think ‘well this isn't going to work’. The time

difference in thinking about all of this stuff ... I don't know, we change our minds all the time.”

The nice thing about using these queries as a conceptualization space over a longer time frame is that even if the purchase is made at some later date or offline, the users still benefited and learned from the guidance that the search provided through the intelligence of the online solutions.

7.3 Competing Solutions

Of course, the online solutions need not necessarily get along. A fundamental assertion of this research has been that the smarter, component-based representations eliminate the need for indexing processes that aren't scalable (see Chapter 3 for more details). Flat representations like Jpegs, DWGs, or Web documents would not have this advantage; they would require an auxiliary RDF index to be generated to make them searchable. This means that maintaining the indexes would not only be time-consuming, but given the current limitations in computer vision, would be something that people would have to do manually. This makes the index generation error-prone and inconsistent, and would probably make them less detailed as well.

When sharing the search space with smarter component systems, these flat solutions would probably get out-competed: they simply wouldn't stay as up-to-date or score as well. But if we take a step back and look at the AEC industry, we might conclude that this actually puts healthy pressure on companies to move away from the traditional representations they've been using for too long. NIST Research has shown that operational inefficiencies, many of which stem from using older representations that are difficult to maintain and share, are the primary reason that the industry has lagged behind most other industries. (Gallagher) The promise of a potentially lucrative service channel for participative design solutions could put positive pressure on builders, developers and manufacturers to move to the new component-based standards that are already taking hold.

7.4 Adding Game-Like Interaction to the Interfaces

As was discussed in the user evaluations in Chapter 5, a very interesting area of inquiry into the conceptual interfaces as a design source would be the incorporation of interactive, multiplayer functionality to make the experience of query-building even more game-like than it already is. The benefit of making things entertaining has already been discussed: we get more detailed queries and better search results from fun interfaces. In observing the couple that came in for this study, it was fascinating to see how the two interacted via the interfaces. The wife would revise some of her husband's tags, and throughout the session they would chat about their selections. In some cases there were disagreements that had to be negotiated, as in "no, the kitchen needs to be open", and in other cases they would cooperate to work through a more difficult decision together. It was also interesting to see how one of them would begin to guide an exercise if they were the more comfortable of the two, particularly in starting out. For example, the husband made the initial Floor Plan Diagram arrangement but when the two were presented with the Activity Sequencer, his wife immediately took the lead. The interfaces are experienced uniquely by different types of problem solvers – interactive interface designs could build upon that.

Designing different mechanisms within the interfaces to enable this sort of interactivity might make the process of searching within the system even *more* educational than it is. Different game-like modes could be explored; for example, an option could be to have a cooperative design game where the first user tries to guess the second user's preferences, after which the second user gets a chance to make revisions to those guesses. In this scenario, the first user might get a score increase based upon the number of revisions that were (or weren't) made by the other. This type of game-like interactivity may help people get more immersed in the exercises and reflect upon things. Additionally, other types of teams than couples could be supported. For example, a user might be paired with a professional designer to work through the exercises; this type of cooperative scenario could certainly be educational for both parties.

7.5 Next steps

This research has shown positive results; as such, the most logical next steps would be to proceed with Web-based implementations that allow for more complete evaluation of the search algorithms. These implementations should also be used for additional user studies, this time with a large number of anonymous Web users, to get a better sense of how the tools function after implementation. This continued development and evaluation would be useful in addressing the limitations of, and problems encountered in the pilot studies discussed here.

While our search analysis had some interesting and positive results, additional and more thorough analysis is needed. A much larger search space needs to be created to give the search algorithm more resources to search through. This is important not only for measuring the system's performance and scalability but also in determining how well the search functions when the search space includes completely different design contexts; in other words, solutions that have much more variance than the ones that were tested thus far.

Additionally, more work needs to be done on the implementation of an actual component system interface, or on the integration with industry component specifications like BIM or IFC. This exploration of BIM systems was outside of the focus of this work but is critical to the success of a consumer-oriented home design system like the one detailed in Chapter 6. There is some existing work that may be followed in this area. T.J. McLeish (MIT House_n) examined the potential of a tangible, tactile interface that allowed for the manipulation of actual 3D objects that were mapped to component system definitions. And Xiayou Ma (MIT House_n) performed an initial analysis of kitchen design typologies that might be useful in a component-based interface aimed at non-expert consumers, for more precise selection algorithms. Additionally, research should be done into the data structure of industry component-based standards to see how well they map to the component system that was developed here (Appendix 2, A2.1). As was previously noted, the analysis here was based upon a SQL-based component specification; this ought

to generalize well to other formats, but this assertion needs to be evaluated.

Additionally, alternative query interfaces could be developed, taking advantage of the standardized RDF query structure as a strategy for providing consumers with different ways to approach architecture. This might be an avenue that architects could explore themselves, investigating for example, innovative search interfaces that allow for the exploration of their own inventory of designs.

7.6 The Future Architect

The complexion of architecture is changing, and the role of the architect will continue to evolve. These days, architects don't have much to do with home design in the United States; they are a non-essential player in a process generally driven by developers. In a process like the one described here, one that is driven by consumers, architects have a new and vital purpose. By providing designs in a component system like the one we've examined, it is the architect that gives the system its most critical element: the logical source for the design guidance that is given to the user. Ironically, many architects seem to be resistant to ideas like the ones behind this work, ideas that, if realized, would provide them with new service channels and therefore new business and design opportunities. Maybe this is due to a tendency to focus upon academic or theoretical in-fighting. Or maybe they feel like working in this sort of system would limit their expressiveness, or maybe they are intimidated by it. But to survive, the architect needs to evolve. Probably, once a system like this begins to take hold and proves to be effective, architects will have a bit more interest in exploring the potentiality of computation to change and improve upon their profession, beyond the narrow confines of seductive representations or the intransigent support of speculative design philosophies. In its purest and most noble form, architecture endeavors to improve, through sense and sensibility the process and form of buildings. And that is what this work is all about.

Appendices

Appendix 1 *User Exercises*

A1.1 Summary of Volunteer Users

	Name	Gender	DOB	Background	Date	Source
1	Thea	F	1969	N/A	4/26/07	Mailing List
2	Jane	F	1965	Affordable Housing Development	4/20/07	Advertisement
3	Jack	M	1965	Information Technology	4/20/07	Advertisement
4	Zahra	F	1979	Student	4/20/07	Mailing List
5	Jeffrey	M	1947	Retired, Software Tester	4/23/07	Advertisement
6	Alison	F	1976	Student	4/23/07	Advertisement
7	Kevin	M	1972	N/A	4/25/07	Advertisement
8	Megan	F	1980	Student	4/25/07	Advertisement
9	Marissa	F	1981	Student, Engineering	5/1/07	Advertisement
10	Liv	F	1974	Student	5/2/07	Mailing List
11	John	M	1942	Retired Librarian	5/2/07	Advertisement
12	Brittany	F	1982	Student, Communications	5/4/07	Advertisement

A1.2 Questionnaire Given to Volunteer Users

MIT House_n Consortium | **home design study** | questionnaire

Name: _____

Date: _____

A. Questionnaire about aptitude with computers and application familiarity.

Please answer a few questions about yourself and your familiarity with various types of computer interfaces. You may skip any question you don't want to answer or feel unable to answer. If you would like any additional details about the subject matter, please don't hesitate to ask.

1. **What is your year of birth?** _____
2. **Have you ever worked with an architect to build or remodel a home?**
3. **How comfortable are you with reading an architectural floor plan?**
4. **How comfortable are you with using a computer**
 - a. I'm a power user; I use one several hours a day, at least
 - b. I'm a regular user, I work with computers daily
 - c. I am comfortable with computers but don't use them at work
 - d. I am not very comfortable with computers, or I'm a novice user
 - e. I rarely use computers, if at all.
 - f. I don't know
5. **What best describes your level of comfort with trying out new programs or utilities on your computer**
 - a. I actively try out new programs or utilities to increase my productivity or learn new skills
 - b. I tend to follow a routine, and use the same applications, but I'm open to new applications or routines when I stumble onto them
 - c. I tend to follow a routine, I'll generally only change my usage pattern if I'm forced to, like with upgrades or through training at work
 - d. I know how to do a few specific tasks but don't have a lot of confidence when it comes to learning new things, so I tend to avoid change
 - e. I am not comfortable with unfamiliar programs or unclear prompts from a computer, these things tend to make me nervous.
 - f. I don't know

A1.2 Questionnaire Given to Volunteer Users

6. What motivates you to learn new applications? For each motivation, please rank from 0-3 where 0 = not a motivation, 1 = weak motivation, 2 = average motivation, and 3 = strong motivation.

- a. ___ Gaining access to new functionality that I haven't had access to
- b. ___ Increasing my productivity
- c. ___ Staying up-to-date with the latest trends or developments
- d. ___ Entertainment value, leisure
- e. ___ Someone at work tells me to
- f. ___ The perception of other benefits
- g. ___ Other: _____

What applications do you use regularly?

(a listing of common applications is provided for the subject to check off)

<i>For each of the applications listed in this chart, please indicate your level of familiarity, if any. If you're uncertain about one of the types listed below, you may check "N/A" and/or ask the interviewer for more information.</i>	Level of Familiarity					
	1 = not familiar, never used 2 = vaguely familiar 3 = familiar, can get around 4 = proficient 5 = expert, power user					
Application Type	1	2	3	4	5	N/A
Using tools like iTunes to view, search for, and manage files						
Using a web browser like Internet Explorer to access web sites						
Web browsing: using search engines like google or yahoo						
Using visual searches like Google Earth , or mapquest .						
Email applications to send and sort emails						
Text editors like Word to create/edit documents						
Presentation software like Powerpoint to create documents with graphic content						
Professional Document management tools like Adobe Illustrator or InDesign						
Graphics editing software like Adobe Photoshop or Paint Shop Pro to manipulate image files: adjusting contrast, size, or format						
Graphics editing software like Photoshop or Paint Shop Pro to draw new images.						
Development environments like Dreamweaver or Flash						
Development environments like .NET , Visual Studio , Java						
CAD tools like AutoCAD , Rhino						
Professional Rendering or animation tools like 3dStudioMax , Maya , or Lightscape						
3D Video Games, either on PC or Console systems, like Halo or Counterstrike						
Massive Multiplayer Video Games like Everquest , or World of Warcraft						
Online, Social Games like Second Life						
Constructive Video Games like Sim City						
Other:						

A1.2 Questionnaire Given to Volunteer Users

Rating 1 = strongly disagree 2 = disagree 3 = agree 4 = strongly agree	Component Assembly	Option Checklist	Activity Sequence	Diagramming the Plan	Packing Metaphor
The arrangement that I made reflected my own needs and values.					
The choices I made during this exercise were very clear to me, nothing was vague or ambiguous, and nothing was confusing					
This exercise allowed me to think of things about my home needs that may not have occurred to me otherwise					
I learned something new that I had never thought about before when doing this exercise					
The exercise was fun					

A1.2 Questionnaire Given to Volunteer Users

General Questions

1. *The conceptual exercises helped me think about my home needs in ways that the first two exercises didn't.*

True / False

Details: _____

2. *The conceptual exercises allowed me to discover or think of home needs more so than the other exercises.*

True / False

3. *In the assembly exercise, what was the most useful representation for you as you selected options?*

4. *In the second exercise, where checklists were given, were the descriptions ambiguous or hard to compare?*

Yes / No

5. Rank the Exercises.

Under each heading, please rank the exercises as far as you can, from best to worst by giving each exercise a specific number from 1 to 5.

Ranking:

1 = Best

5 = Worst

Exercise	Ease of use	Helped you learn something	Overall Impression
Component Assembly			
Option Lists			
Activity Sequencer			
Diagramming			
Packing Metaphor			

A1.3 Library Cards For Component Assembly Exercise

Exercise designed by K Larson, M G Phillips, and C Farina, card layouts by C Farina & M G Phillips

<p>3D_07 Bedroom & Study INCLUDE bedroom for large study space</p>	<p>3D_08 Bedroom / Study LARGE room for study in bedroom, with study desk</p>	<p>3D_15 Study LARGE room for study study desk type</p>	<p>3D_17 Living LARGE living area study, reading & TV space</p>	<p>3D_19 & 2D_08 Study & Work LARGE room area for STUDY, reading study</p>
BEDROOM & STUDY				
<p>3D_02 & 2D_13 Bedroom & Study INCLUDE 1 bedroom study desk</p>	<p>3D_02 & 2D_09 Bedroom & Study INCLUDE bedroom with study desk</p>	<p>3D_03 & 2D_10 Bedroom & Study INCLUDE bedroom with study desk</p>	<p>3D_05 & 2D_03 Bedroom & Study INCLUDE bedroom with study desk</p>	<p>3D_06 Bedroom & Study INCLUDE bedroom study desk</p>
BEDROOM & STUDY				

A1.3 Library Cards For Component Assembly Exercise

BATHROOM					BATHROOM		
							
<p>ID_01</p> <p>Second Bathroom SMALL, single bathroom with shower and toilet</p>	<p>ID_02</p> <p>Second Bathroom SMALL, single bathroom with shower and toilet and toilet</p>	<p>ID_03</p> <p>Second Bathroom SMALL, bathroom with shower and toilet and toilet</p>	<p>ID_04</p> <p>Second Bathroom & SHOWER SMALL, bathroom with shower and toilet and toilet</p>	<p>ID_05</p> <p>Second Bathroom MEDIUM, bathroom with shower & toilet and toilet</p>	<p>ID_07</p> <p>Second Bathroom Double bathroom, large with shower & toilet and toilet</p>	<p>ID_08</p> <p>Second Bathroom Two large bathrooms, large with shower, toilet and toilet</p>	<p>ID_09</p> <p>Second Bathroom MEDIUM, bathroom with shower, and toilet and toilet for the entry</p>
							

A1.3 Library Cards For Component Assembly Exercise

KITCHEN & ENTRY

IB_01 & 1C_01 Kitchen Dining & Entry Lift shaft Lift shaft Lift shaft	IB_01 & 1C_01	IB_03 & 1C_06 Kitchen Dining & Entry Lift shaft Lift shaft Lift shaft	IB_05 & 1C_03 Kitchen & Dining Entry Lift shaft Lift shaft Lift shaft	IB_06 & 1C_08 Kitchen & Entry + Entry DOUBLE DOORS	
IB_07 & 1C_04 Entry Kitchen & Dining Lift shaft Lift shaft	IB_08 & 1C_11 Entry Kitchen & Dining Lift shaft Lift shaft Lift shaft	IB_11 & 1C_04 Kitchen & Entry Lift shaft Lift shaft	IB_12 & 1C_04 Kitchen & Entry Dining CLOSED	IB_13 & 1C_07 Kitchen & Entry Dining CLOSED with door	

LIVING & DINING

<p>3B_01 & 3C_17</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_06 & 3C_17</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>1B_01 & 1C_18</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_04 & 3C_03</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_03 & 3C_13</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>
<p>3B_14 & 3C_17</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_18 & 3C_18</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_22 & 3C_22</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_20 & 3C_20</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>	<p>3B_23 & 3C_23 & 3D_23</p> <p>Living & Dining with fireplace, wood paneling, and built-in shelving.</p>






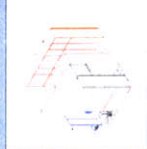
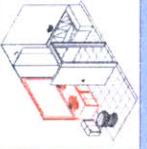

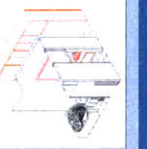
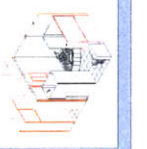




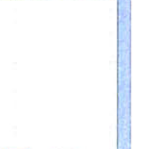



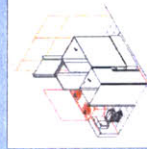
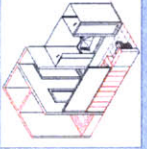

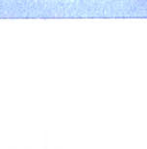

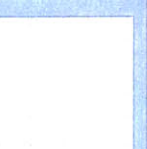
LIVING & DINING

A1.3 Library Cards For Component Assembly Exercise

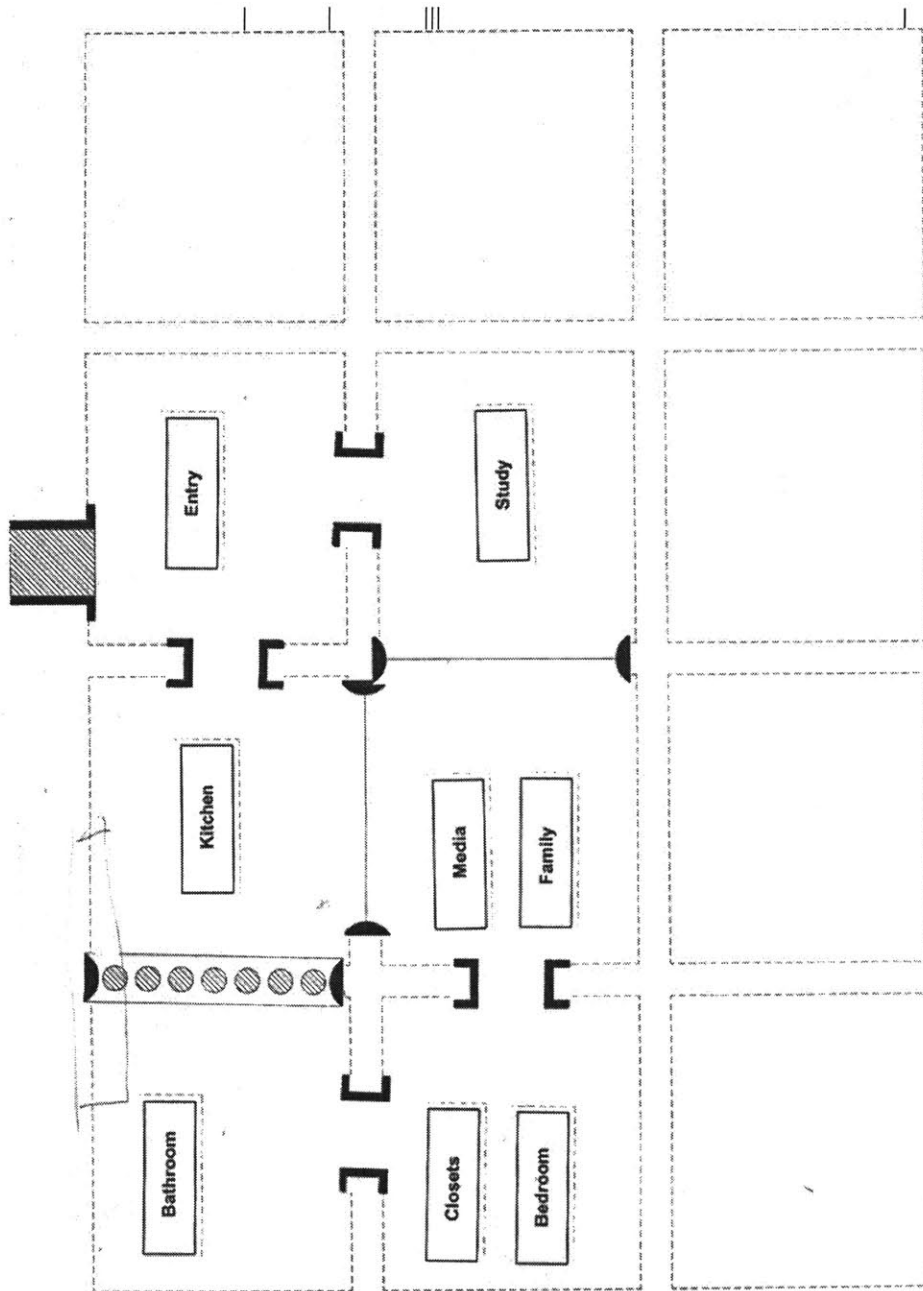
A1.3 Library Cards For Component Assembly Exercise

  3A_04 & 2A_07 LARGE Master Bedroom with Master Closet and Master Bathroom	  3A_09 & 2A_01 LARGE Master Bedroom with Master Closet	  3A_10 & 2A_03 MEDIUM Master Bedroom with Master Closet and Master Bathroom	  3A_11 LARGE Master Bedroom with Master Closet and Master Bathroom	  3A_12 & 2A_01 LARGE Master Bedroom with Master Closet
 3A_01 & 2A_06 SMALL Master Bedroom with Master Closet and Master Bathroom	 3A_02 & 2A_08 MEDIUM Master Bedroom with Master Closet and Master Bathroom	 3A_07 & 2A_02 MEDIUM Master Bedroom with Master Closet	 3A_07 MEDIUM Master Bedroom with Master Closet	 3A_05 & 2A_08 MEDIUM Master Bedroom with Master Closet and Master Bathroom
MASTERBEDROOM & MASTERCLOSET				
  3A_01 & 2A_06 SMALL Master Bedroom with Master Closet and Master Bathroom	  3A_02 & 2A_08 MEDIUM Master Bedroom with Master Closet and Master Bathroom	  3A_07 & 2A_02 MEDIUM Master Bedroom with Master Closet	  3A_07 MEDIUM Master Bedroom with Master Closet	  3A_05 & 2A_08 MEDIUM Master Bedroom with Master Closet and Master Bathroom
MASTERBEDROOM & MASTERCLOSET				

A1.3 Library Cards For Component Assembly Exercise

MASTER BATHROOM				
				
				
<p>JA_01 & ZA_11 Bathroom & Closet Includes Master Bedroom, Master Bathroom, and Walk-In Closet (WIC)</p>	<p>JA_02 Master Bedroom Small Single Master Bedroom, Master Bathroom, and WIC</p>	<p>JA_04 & ZA_09 Bedroom & Closet Includes Master Bedroom, Master Bathroom, and Small Master Closet</p>	<p>JA_05 & ZA_14 Bedroom & Closet Includes Master Bedroom, Master Bathroom, and Small Master Closet</p>	<p>JA_07 & ZA_04 Bedroom & Closet Includes Master Bedroom, Master Bathroom, and Small Master Closet</p>
				
<p>JA_09 Master Bedroom Includes Master Bedroom, Master Bathroom, and WIC</p>	<p>JA_10 Master Bedroom Large Master Bedroom, Master Bathroom, and WIC</p>	<p>JA_11 Master Bedroom & Large Master Bedroom, Master Bathroom, and WIC</p>		
				
				
				
MASTER BATHROOM				

A1.4 Example Given for Floor Plan Diagram Exercise



A1.5 Photographs of User Exercises

Activity Sequencer

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date: _____
Session: _____
Task #: _____
Sheet #: _____

1. Activities: List the specific activities for this sequence, in chronological order.
Example: if you were listing out the activities of your morning routine, you might start with "wake up" followed by "take shower" and so on.

2. Room: Here, list the room in which each activity takes place.

3. Transition between activities: How does your position or orientation change between the activities?

4. Reflecting on the sequence here, what qualities, styles, or items are important for you to have included in this room in a new home?

Exercise: Activity Sequencer

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date: _____
Session: _____
Task #: _____
Sheet #: _____

1. Activities: List the specific activities for this sequence, in chronological order.
Example: if you were listing out the activities of your morning routine, you might start with "wake up" followed by "take shower" and so on.

2. Room: Here, list the room in which each activity takes place.

3. Transition between activities: How does your position or orientation change between the activities?

4. Reflecting on the sequence here, what qualities, styles, or items are important for you to have included in this room in a new home?

Exercise: Activity Sequencer

A1.5 Photographs of User Exercises

Activity Sequencer

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date: _____
Session: _____
Task #: _____
Sheet #: _____

1. Activities: List the specific activities for this sequence, in chronological order.
Example: if you were listing out the activities of your morning routine, you might start with "wake up" followed by "take shower" and so on.

	Snoozing	Waking Up	Showering	Brush Teeth	Check Weather	Dressing
2. Room: Here, list the room in which each activity takes place	Bedroom	Bedroom	Bathroom	Bathroom	Bedroom	Bedroom
3. Transition between activities: how does your position or orientation change between the activities?		Change in Position	Move Through Multiple Spaces	Move to a Different Part of Room	Move Through Multiple Spaces	Move to a Different Part of Room
4. Reflecting on the sequence here, what qualities, styles, or items are important for you to have included in this room in a new home?	Spacious Private	Abundant Natural Light	Private Lots of Counterspace Abundant Artificial Light			Lots of Counterspace

Exercise: Activity Sequencer

MIT House_n / Changing Places
Open Source Building Alliance: Conceptualization Tools

Date: _____
Session: _____
Task #: _____
Sheet #: _____

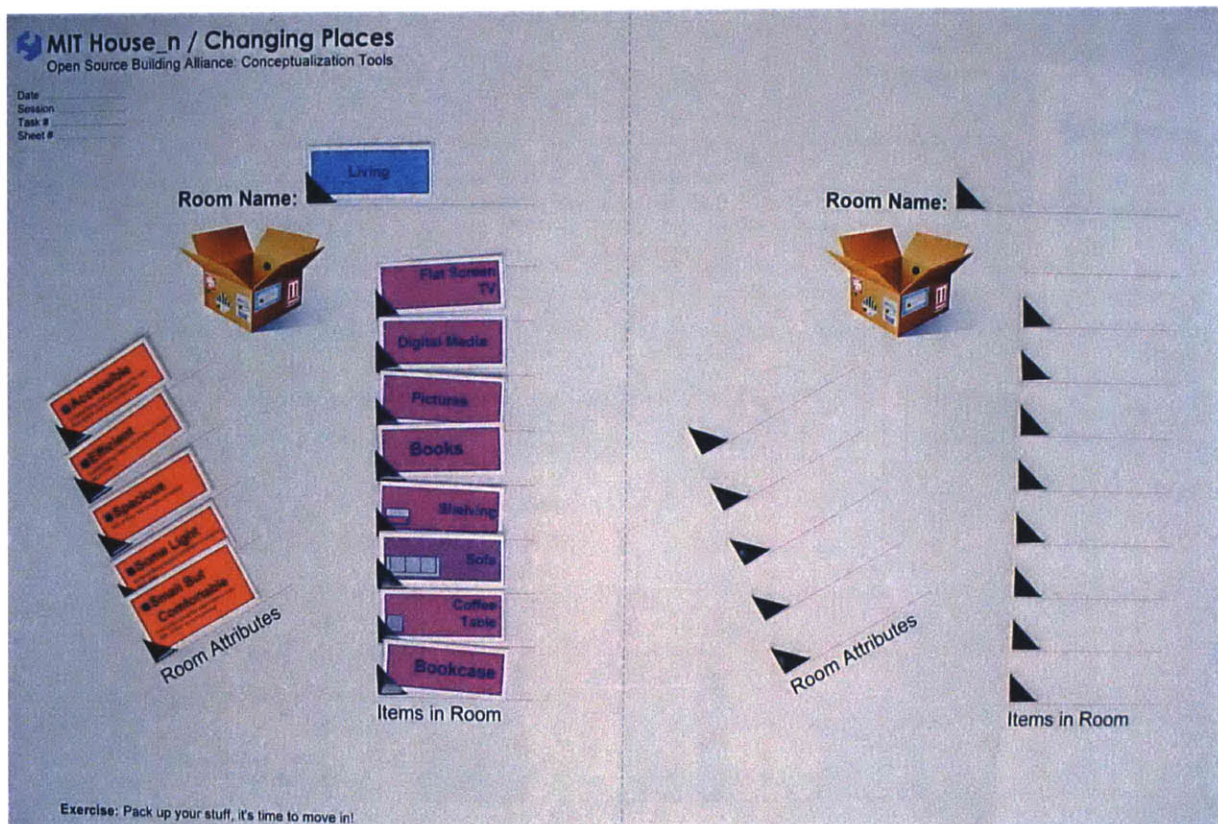
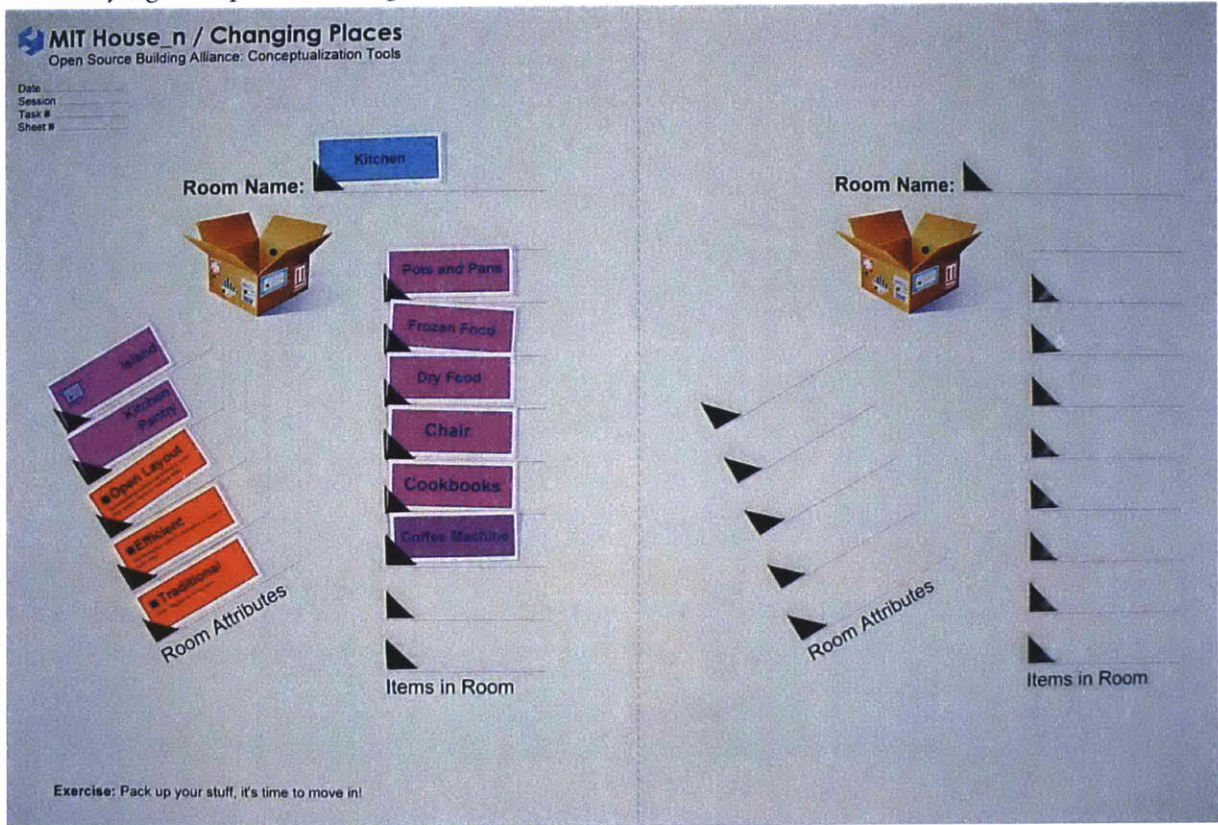
1. Activities: List the specific activities for this sequence, in chronological order.
Example: if you were listing out the activities of your morning routine, you might start with "wake up" followed by "take shower" and so on.

	Waking Up	Check Weather	Brush Teeth	Showering	Reading	Computer Use
2. Room: Here, list the room in which each activity takes place	Bedroom	Bedroom	Bathroom	Bathroom	Living	Bedroom
3. Transition between activities: how does your position or orientation change between the activities?		No Change in Position / Orientation	Move Through Doorway	Move to a Different Part of Room	Move Through Multiple Spaces	Move Through Multiple Spaces
4. Reflecting on the sequence here, what qualities, styles, or items are important for you to have included in this room in a new home?	Telephone Abundant Natural Light Small But Comfortable	Flat Screen TV Private Bed	Shower Toilet	Bathroom Sink Private	Clothing Knick Knacks Contemporary Shelving	Office Desk Chair Accessible

Exercise: Activity Sequencer

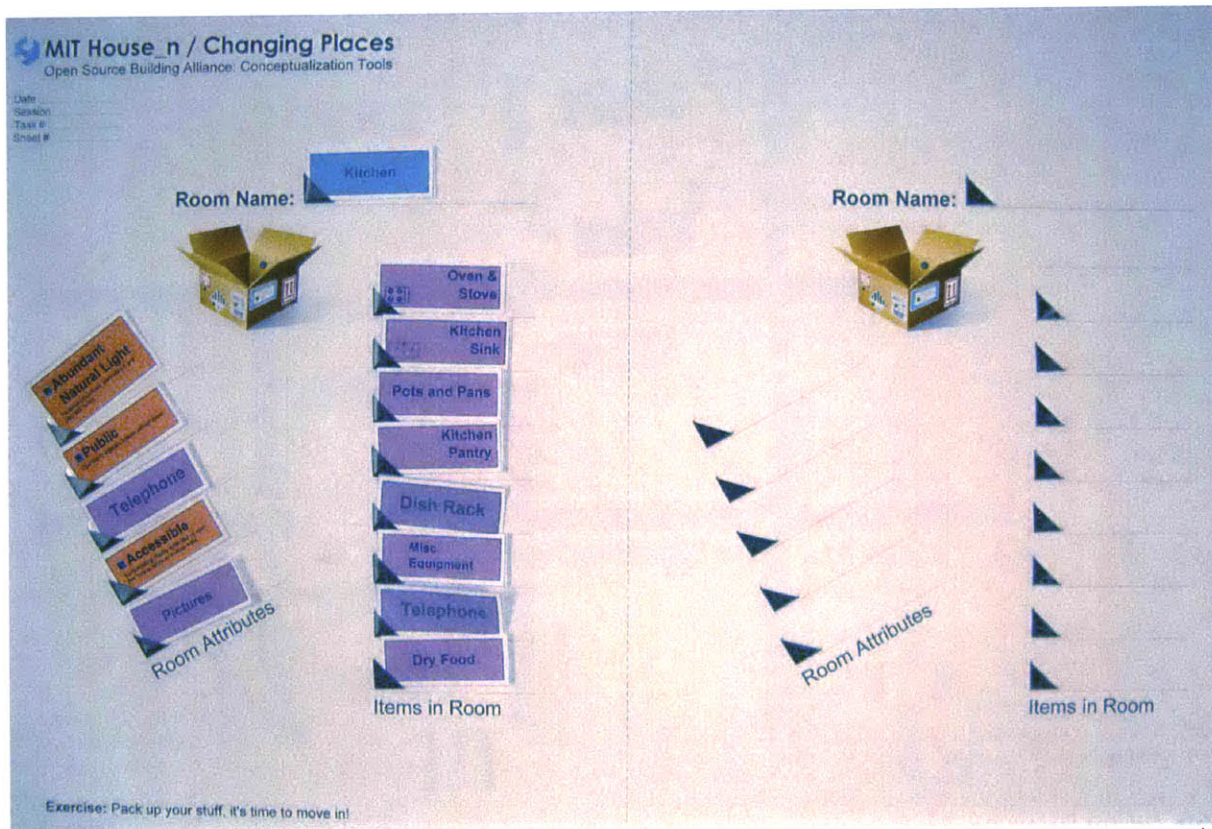
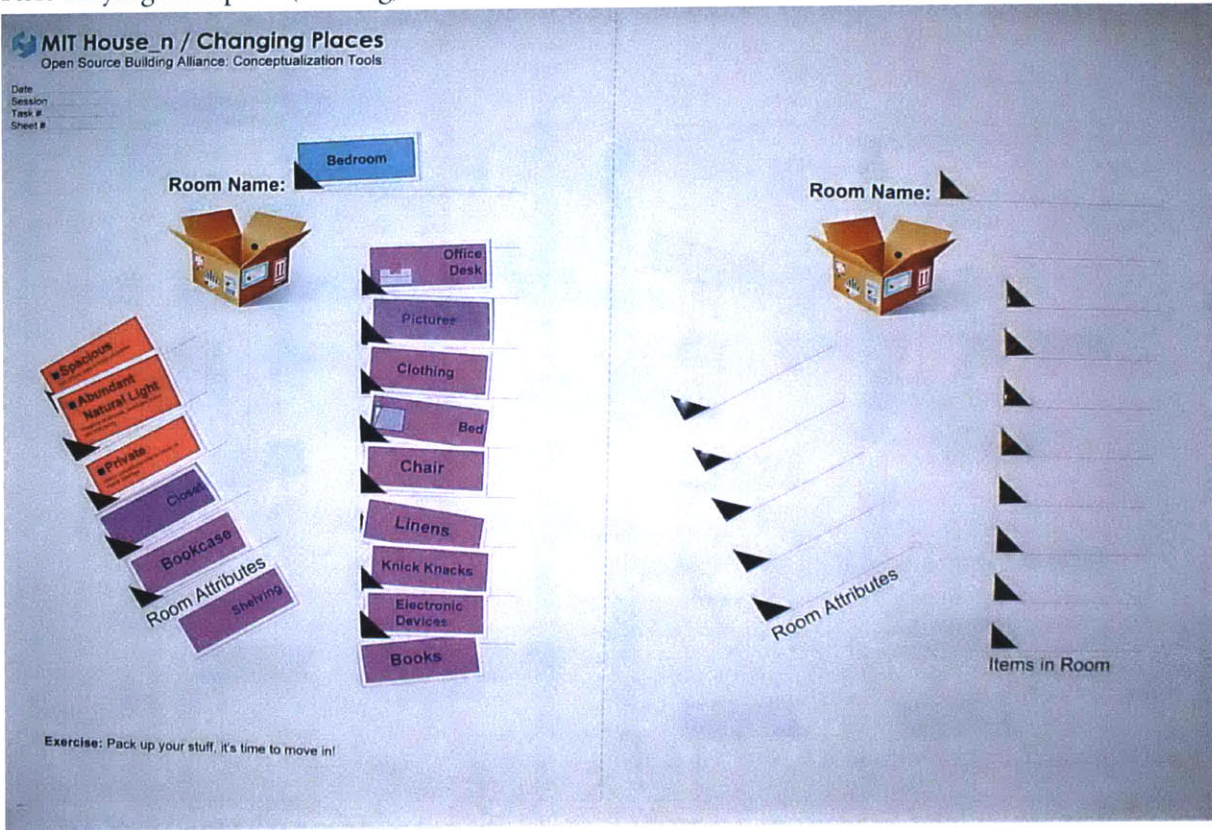
A1.5 Photographs of User Exercises

Role-Playing Metaphor (Packing)



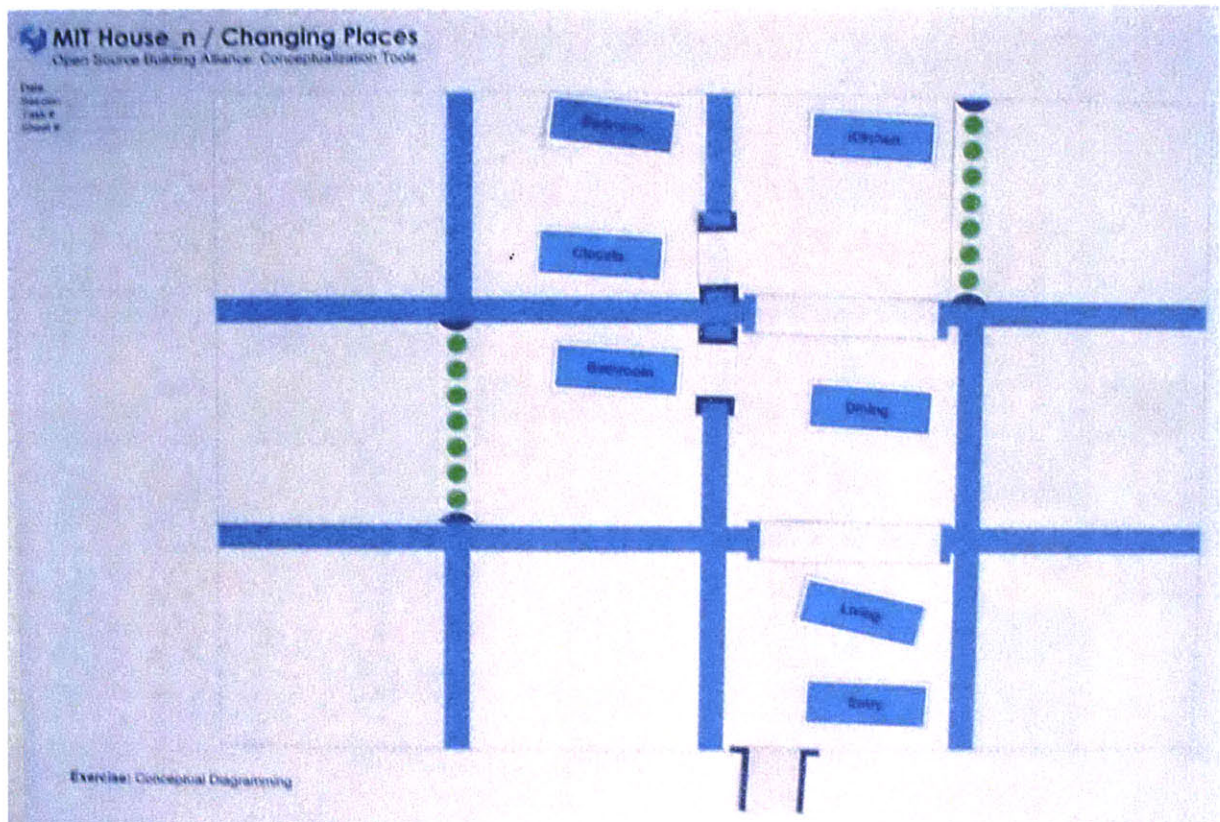
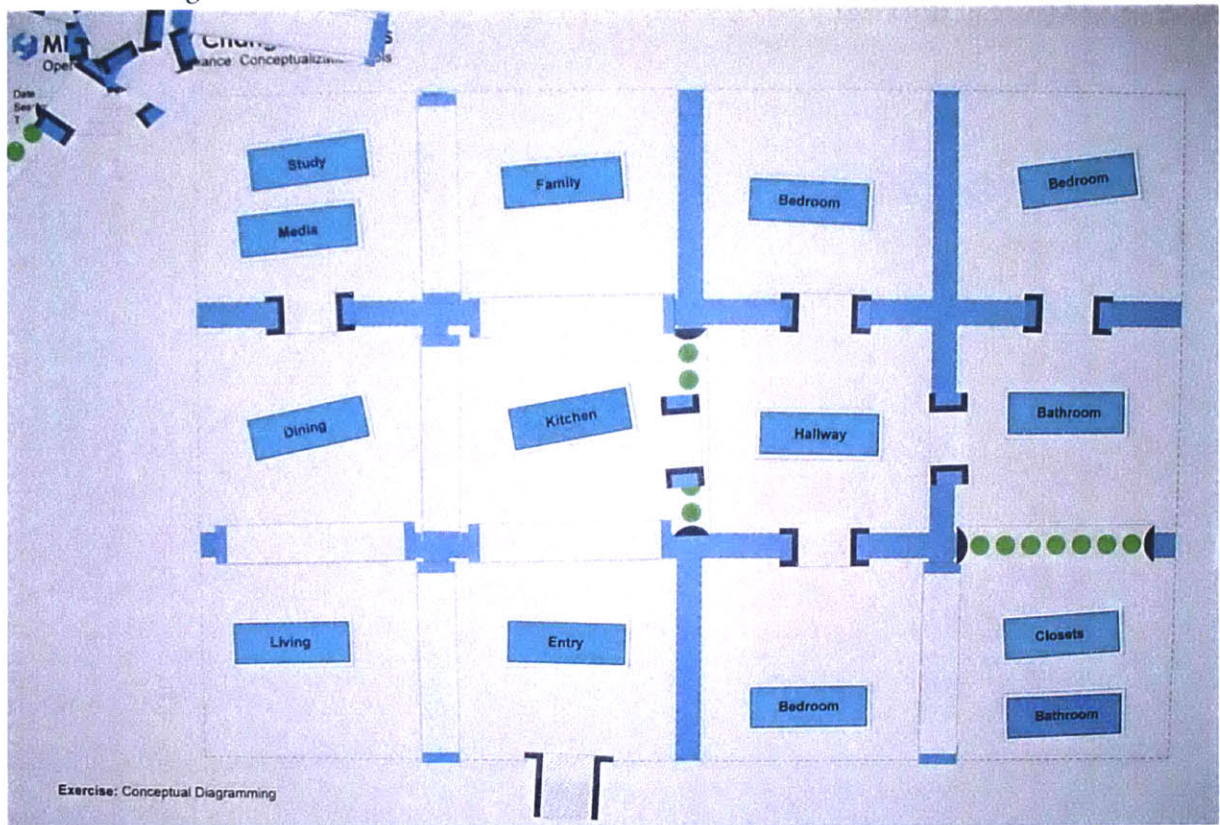
A1.5 Photographs of User Exercises

Role-Playing Metaphor (Packing)



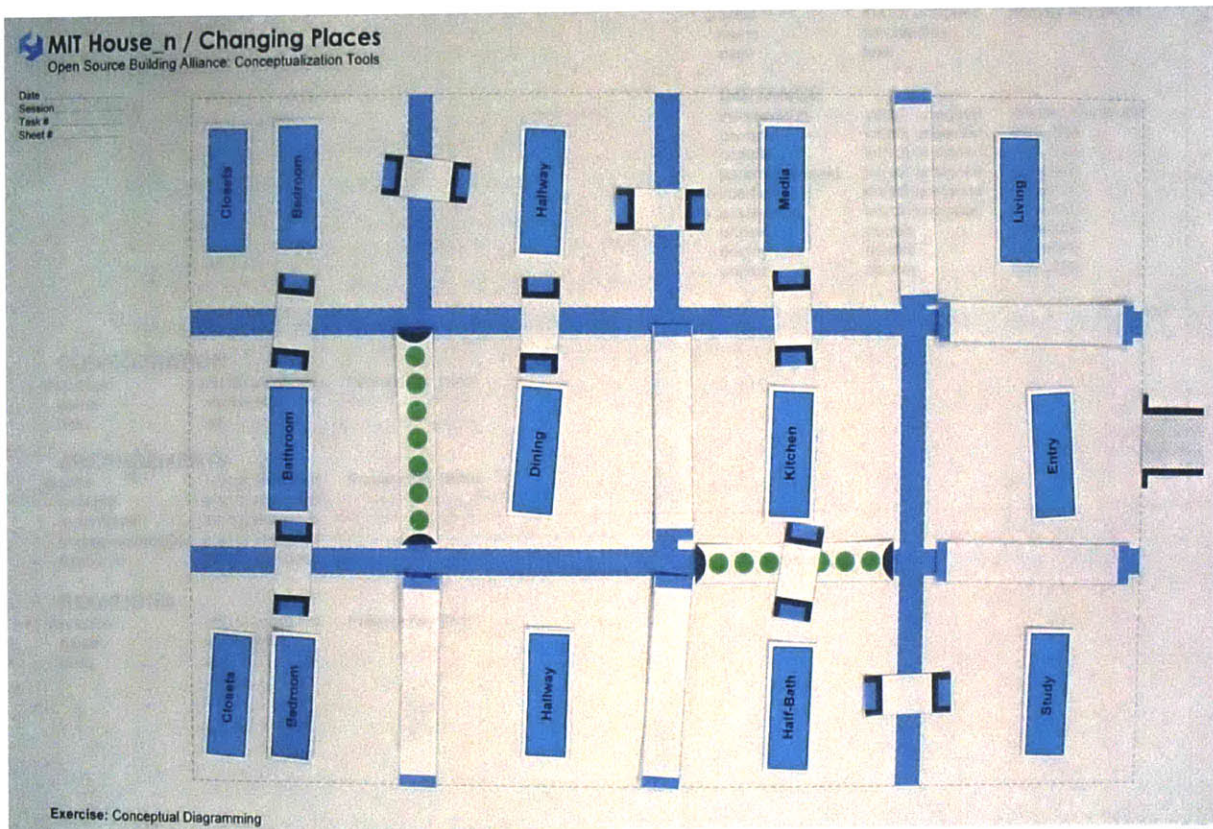
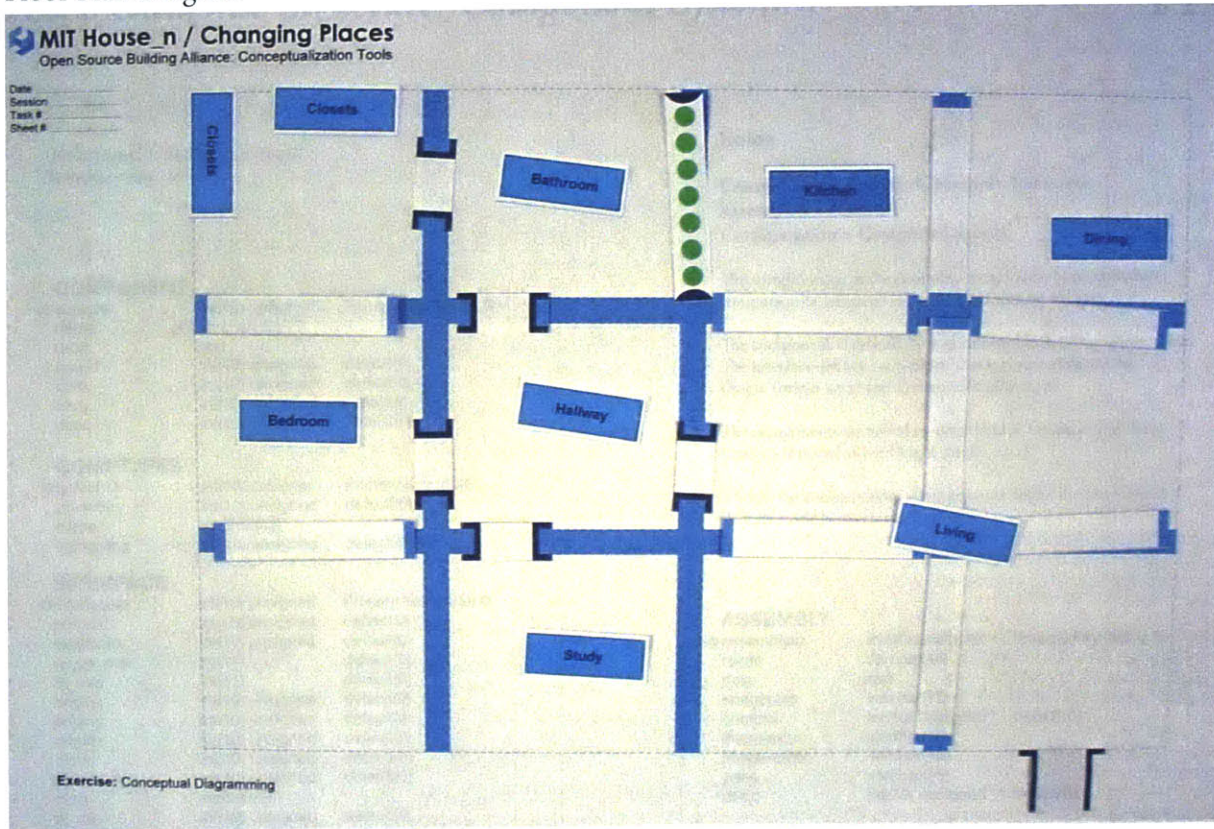
A1.5 Photographs of User Exercises

Floor Plan Diagram



A1.5 Photographs of User Exercises

Floor Plan Diagram

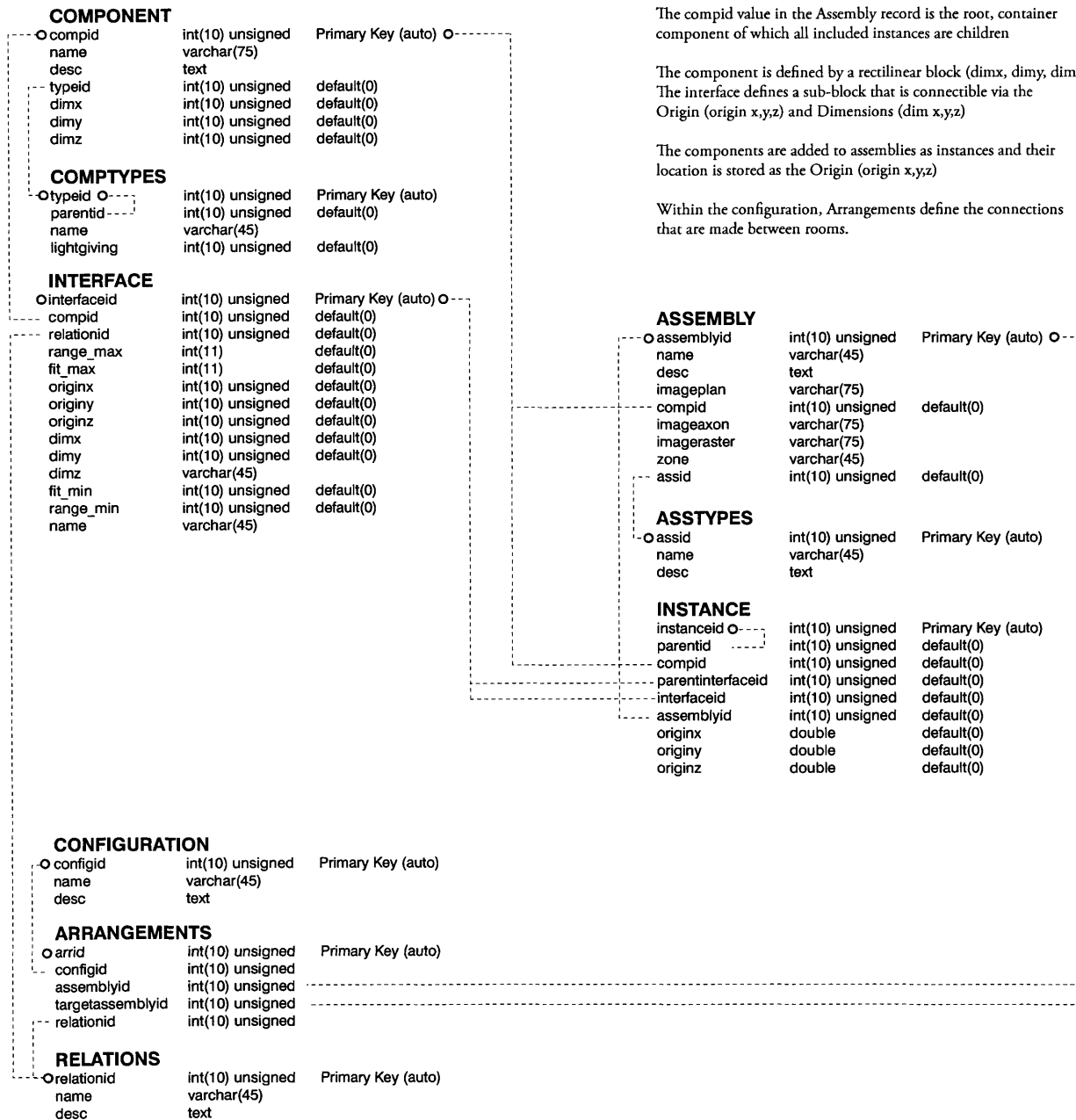


Appendix 2 *System Details*

A2.1 Database Structure: Component System

Relational Database Schema

○ Primary Key



Notes

Components = Furniture, Fixtures, Items, etc.

Assemblies = Rooms

Configuration = Complete Layouts

The compid value in the Assembly record is the root, container component of which all included instances are children

The component is defined by a rectilinear block (dimx, dimy, dimz)
The interface defines a sub-block that is connectible via the Origin (origin x,y,z) and Dimensions (dim x,y,z)

The components are added to assemblies as instances and their location is stored as the Origin (origin x,y,z)

Within the configuration, Arrangements define the connections that are made between rooms.

A2.2 Program Structure: Search Algorithm

Search Application Schema

SearchDesignerCore contains the core search algorithm:

1. Connects to Component Database
2. Invokes calls to read and write RDF
3. Performs search (each stage)
4. Assigns Scores and Sorts

SearchDesignerCore

Methods

```
SearchDesignerCore.AddIndex()
SearchDesignerCore.AddScore()
SearchDesignerCore.CheckForAssembly()
SearchDesignerCore.CheckScore()
SearchDesignerCore.Clear()
SearchDesignerCore.DbConnect()
SearchDesignerCore.DbDisconnect()
SearchDesignerCore.GetResults()
SearchDesignerCore.MySqlExecQuery()
SearchDesignerCore.return_results()
SearchDesignerCore.Search()
SearchDesignerCore.SearchCheckAttributes()
SearchDesignerCore.SearchCheckRooms()
SearchDesignerCore.SearchDesignerCore()
SearchDesignerCore.SearchForConfig()
SearchDesignerCore.SortAndSummary()
```

Properties

```
SearchDesignerCore.ErrorString
SearchDesignerCore.FileName
SearchDesignerCore.HasResults
SearchDesignerCore.Graph
SearchDesignerCore.Query
SearchDesignerCore.RDFParser
```

The below classes allow the system to associate specific relevance scores with each of the values it reads from the incoming query:

SdcInt

```
SdcInt.Value
SdcInt.Score
```

SdcString

```
SdcString.Uid
SdcString.Value
SdcString.Score
```

SdcScore

```
SdcScore.CompareTo(object)
SdcScore.AssemblyID
SdcScore.ConfigID
SdcScore.Score
```

The Graph Stores RDF data in raw arc (triple) format

SearchDataArc

Methods

```
SearchDataArc.SearchDataArc()
```

Properties

```
SearchDataArc.Predicate
SearchDataArc.Resource
SearchDataArc.ResourceUid
SearchDataArc.RObject
SearchDataArc.RobjectId
```

SearchDataGraph

Methods

```
SearchDataGraph.AddArc()
SearchDataGraph.NormalizeGraph()
SearchDataGraph.SearchDataGraph()
```

Properties

```
SearchDataGraph.Graph
(List) Arcs
```

Stores all properties for each room

SearchDataZone

Methods

```
SearchDataZone.SearchDataZone()
```

Properties

```
SearchDataZone.Comments
SearchDataZone.Name
SearchDataZone.Size
SearchDataZone.Uid
(List) Attributes
(List) Items
(List) Connections
(List) Activities
```

Stores the query values for processing

SearchDesignerQuery

Methods

```
SearchDesignerQuery.FindAssembly()
SearchDesignerQuery.FindConfig()
SearchDesignerQuery.FindConfigCompl()
SearchDesignerQuery.FindValue()
SearchDesignerQuery.SearchDesignerQuery()
```

Properties

```
SearchDesignerQuery.Comments
SearchDesignerQuery.Name
SearchDesignerQuery.PredicateInt
SearchDesignerQuery.PredicateIntCompl
SearchDesignerQuery.PredicateStr
(List) Zones
```

Reads the incoming query RDF and writes the output query (with score details)

SearchDesignerRDF

Methods

```
SearchDesignerRDF.ParseQuery()
SearchDesignerRDF.ReplyQuery()
```

A2.3 *The RDF Queries Used for Analysis 1*

Activity Sequencer Source Query

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Bedroom">
    <activity>Waking Up</activity>
    <activity>Check Weather</activity>
    <activity>Dressing</activity>
    <multi>Kitchen</multi>
    <doorway>Bathroom</doorway>
    <attribute>Private</attribute>
    <attribute>Spacious</attribute>
    <attribute>Abundant Natural Light</attribute>
    <item>Shelves</item>
    <item>Closet</item>
  </Description>
  <Description room="Kitchen">
    <activity>Eating</activity>
    <multi>Bedroom</multi>
    <attribute>Lots of Counterspace</attribute>
    <attribute>Spacious</attribute>
    <attribute>Abundant Natural Light</attribute>
    <item>Island</item>
    <item>Table</item>
  </Description>
  <Description room="Bathroom">
    <activity>Grooming</activity>
    <activity>Showering</activity>
    <multi>Kitchen</multi>
    <doorway>Bedroom</doorway>
    <attribute>Private</attribute>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Lots of Counterspace</attribute>
    <item>Shelves</item>
  </Description>
</RDF>
```

A2.3 *The RDF Queries Used for Analysis 1*

Role-Playing Metaphor (Packing) Source Query

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Kitchen">
    <attribute>Contemporary</attribute>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Spacious</attribute>
    <attribute>Accessible</attribute>
    <attribute>Efficient</attribute>
    <attribute>Lots of Counterspace</attribute>
    <attribute>Private</attribute>
    <item>Island</item>
    <item>Pots and Pans</item>
    <item>Shelves</item>
    <item>Refrigerator</item>
    <item>Microwave</item>
    <item>Oven</item>
    <item>Stove</item>
    <item>Table</item>
  </Description>
</RDF>
```

A2.3 *The RDF Queries Used for Analysis 1*

Floor Plan Diagram Query

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Living">
    <function>Closets</function>
    <open>Kitchen</open>
    <open>Study</open>
    <item>Flat Screen TV</item>
    <item>Front Door</item>
    <item>Digital Media</item>
  </Description>
  <Description room="Study">
    <doorway>Kitchen</doorway>
    <open>Living</open>
    <item>Pictures</item>
    <item>Small Coffee Table</item>
    <item>Love Seat</item>
  </Description>
  <Description room="Kitchen">
    <open>Living</open>
    <doorway>Study</doorway>
    <function>Dining</function>
    <item>Island</item>
  </Description>
  <Description room="Bedroom">
    <function>Closets</function>
  </Description>
</RDF>
```

A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

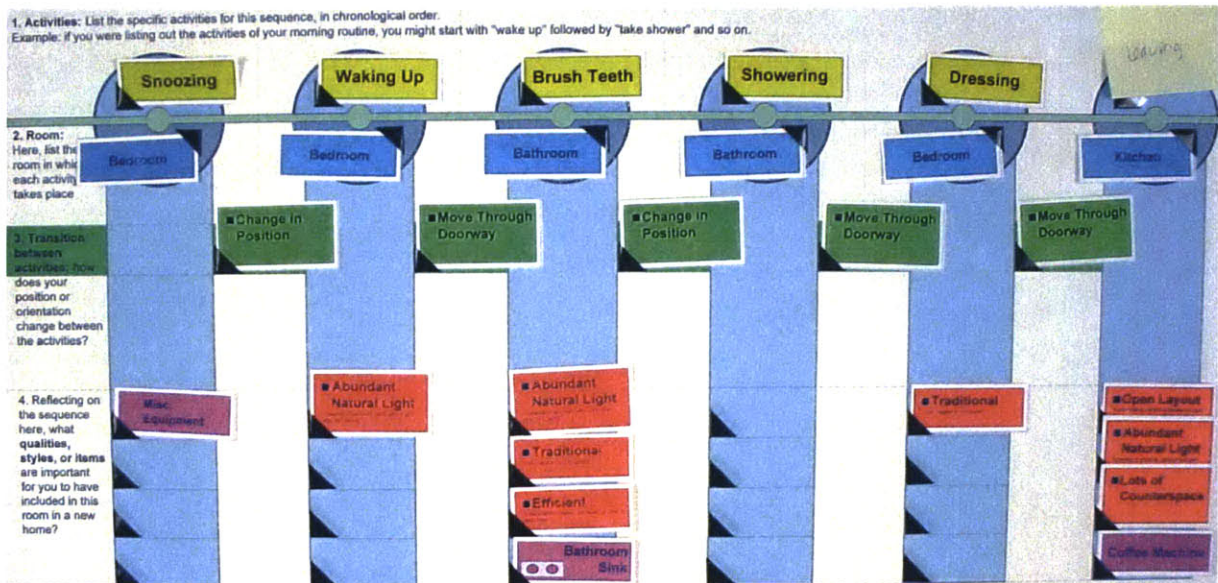
Activity Sequencer 01, by Thea

```

<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Bedroom">
    <activity>Waking Up</activity>
    <activity>Dressing</activity>
    <activity>Snoozing</activity>
    <doorway>Kitchen</doorway>
    <doorway>Bathroom</doorway>
    <attribute>Traditional</attribute>
    <attribute>Abundant Natural Light</attribute>
    <item>Equipment</item>
  </Description>
  <Description room="Kitchen">
    <activity>Leaving</activity>
    <doorway>Bedroom</doorway>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Open Layout</attribute>
    <attribute>Lots of Counterspace</attribute>
    <item>Coffee Machine</item>
  </Description>
  <Description room="Bathroom">
    <activity>Showering</activity>
    <activity>Brush Teeth</activity>
    <doorway>Bedroom</doorway>
    <attribute>Traditional</attribute>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Efficient</attribute>
    <item>Sink</item>
  </Description>
</RDF>

```

Thea's completed exercise:



A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

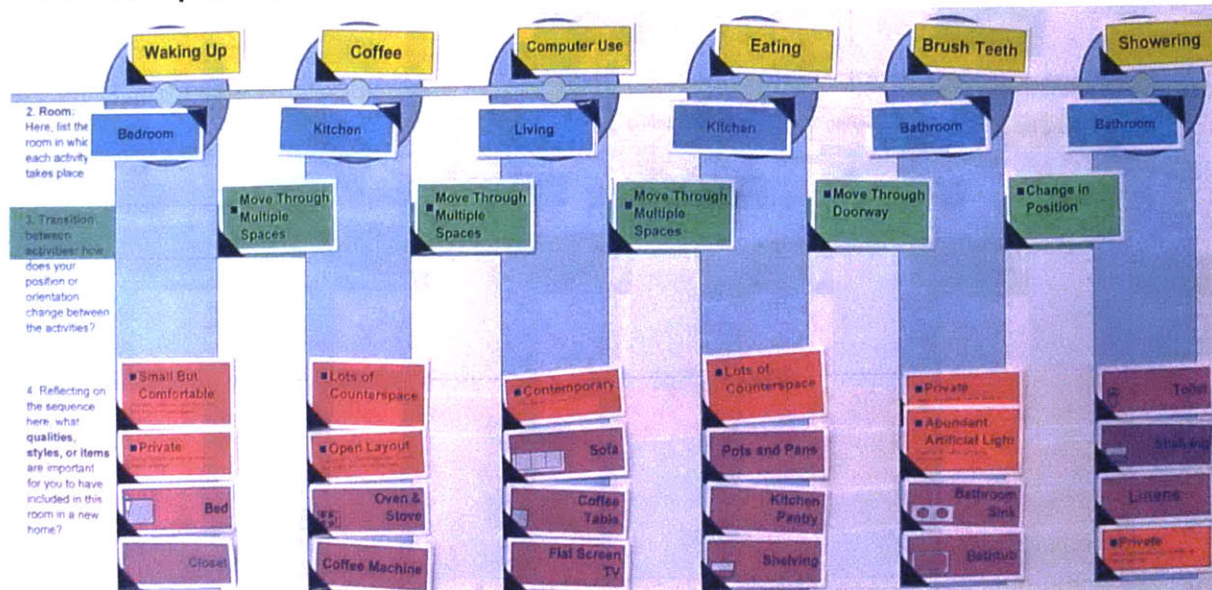
Activity Sequence 2, by Alison

```

<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Bedroom">
    <activity>Waking Up</activity>
    <multi>Kitchen</multi>
    <attribute>Small But Comfortable</attribute>
    <attribute>Private</attribute>
    <item>Bed</item><item>Closet</item>
  </Description>
  <Description room="Kitchen">
    <activity>Coffee</activity>
    <activity>Eating</activity>
    <multi>Bedroom</multi>
    <multi>Living</multi>
    <doorway>Bathroom</doorway>
    <attribute>Open Layout</attribute>
    <attribute>Lots of Counterspace</attribute>
    <item>Oven</item><item>Coffee Machine</item>
    <item>Pots and Pans</item><item>Kitchen Pantry</item><item>Shelves</item>
  </Description>
  <Description room="Bathroom">
    <activity>Showering</activity><activity>Brush Teeth</activity>
    <doorway>Kitchen</doorway>
    <attribute>Private</attribute>
    <attribute>Abundant Artificial Light</attribute>
    <item>Sink</item><item>Bathtub</item>
    <item>Toilet</item><item>Shelves</item><item>Linens</item>
  </Description>
  <Description room="Living">
    <activity>Computer Use</activity>
    <multi>Kitchen</multi>
    <attribute>Contemporary</attribute>
    <item>Sofa</item>
    <item>Coffee Table</item>
    <item>Television</item>
  </Description>
</RDF>

```

Alison's completed exercise:



A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

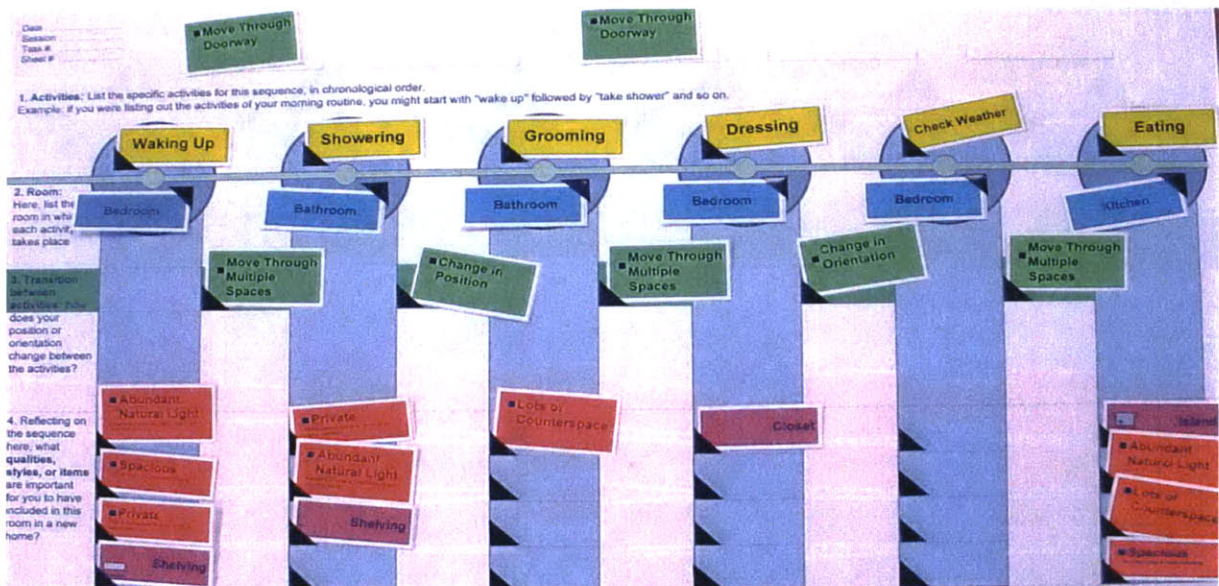
Activity Sequence 3, by Megan

```

<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/" >
  <Description room="Bedroom">
    <activity>Waking Up</activity>
    <activity>Check Weather</activity>
    <activity>Dressing</activity>
    <multi>Kitchen</multi>
    <doorway>Bathroom</doorway>
    <attribute>Private</attribute>
    <attribute>Spacious</attribute>
    <attribute>Abundant Natural Light</attribute>
    <item>Shelves</item>
    <item>Closet</item>
  </Description>
  <Description room="Kitchen">
    <activity>Eating</activity>
    <multi>Bedroom</multi>
    <attribute>Lots of Counterspace</attribute>
    <attribute>Spacious</attribute>
    <attribute>Abundant Natural Light</attribute>
    <item>Island</item>
  </Description>
  <Description room="Bathroom">
    <activity>Grooming</activity>
    <activity>Showering</activity>
    <multi>Kitchen</multi>
    <doorway>Bedroom</doorway>
    <attribute>Private</attribute>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Lots of Counterspace</attribute>
    <item>Shelves</item>
  </Description>
</RDF>

```

Megan's completed exercise:

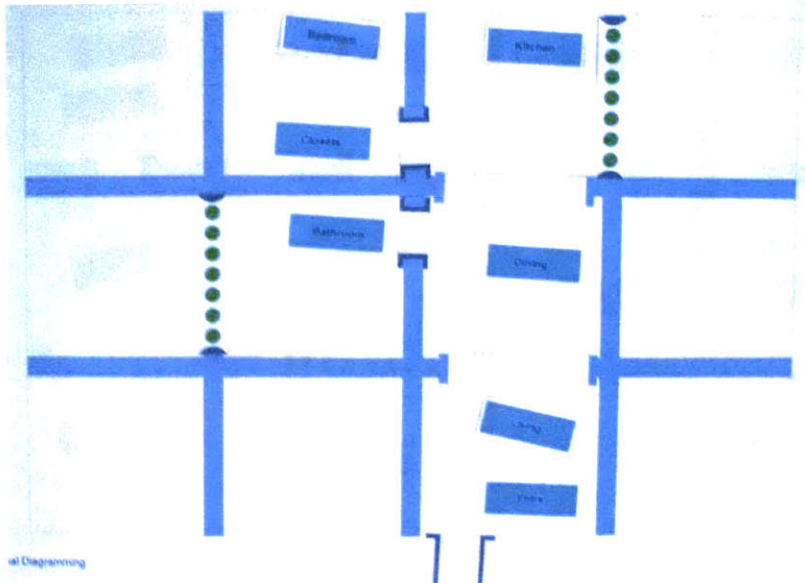


A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Floor Plan Diagram 1, by Liv

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Living">
    <function>Entry</function>
    <open>Dining</open>
    <item>Front Door</item>
  </Description>
  <Description room="Dining">
    <doorway>Bathroom</doorway>
    <open>Living</open>
    <open>Kitchen</open>
  </Description>
  <Description room="Kitchen">
    <open>Dining</open>
    <doorway>Bedroom</doorway>
  </Description>
  <Description room="Bathroom">
    <doorway>Dining</doorway>
  </Description>
  <Description room="Bedroom">
    <function>Closets</function>
    <doorway>Kitchen</doorway>
  </Description>
</RDF>
```

Liv's completed exercise:



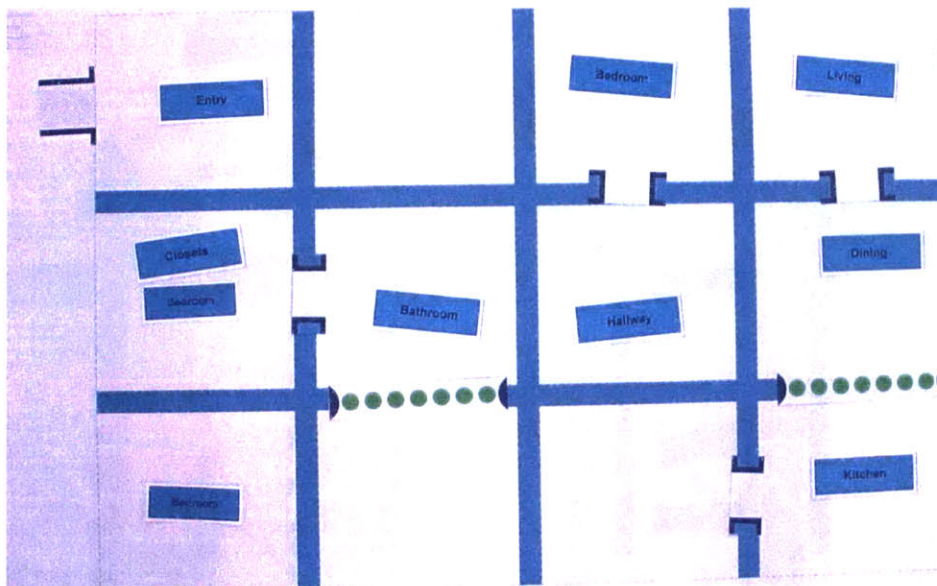
A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Floor Plan Diagram 2, by Zahra

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Entry">
    <function>Entry</function>
  </Description>
  <Description room="Bedroom">
  </Description>
  <Description room="Bedroom" id="02">
    <function>Closets</function>
    <doorway>Bathroom</doorway>
  </Description>
  <Description room="Bathroom">
    <doorway id="02">Bedroom</doorway>
  </Description>
  <Description room="Bedroom" id="03">
    <doorway id="03">Hallway</doorway>
  </Description>
  <Description room="Dining">
    <doorway>Living</doorway>
  </Description>
  <Description room="Living">
    <doorway>Dining</doorway>
  </Description>

  <Description room="Kitchen">
    <doorway></doorway>
  </Description>
</RDF>
```

Zahra's completed exercise:

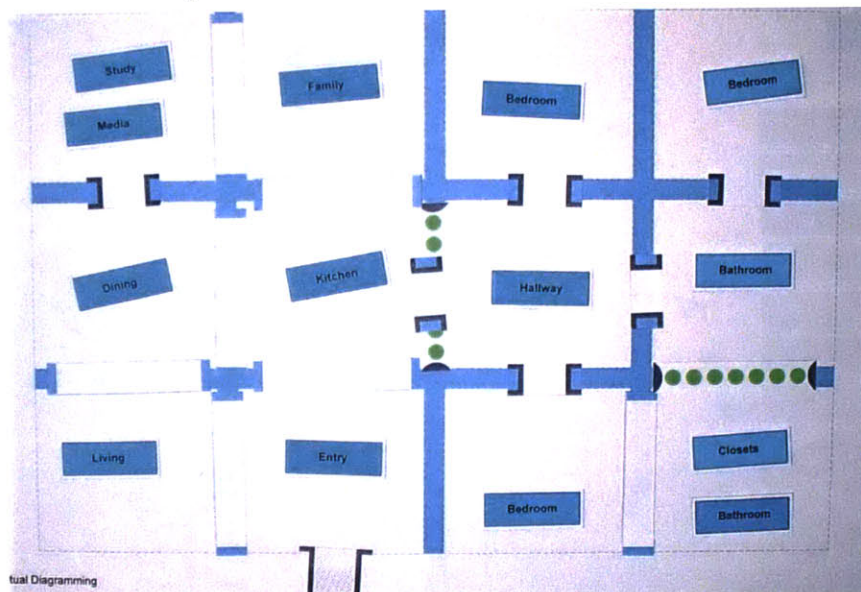


A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Floor Plan Diagram 3, by Marisa

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" dc="http://purl.org/dc/elements/1.1/">
  <Description room="Entry">
    <function>Entry</function> <open>Kitchen</open> <open>Living</open>
  </Description>
  <Description room="Living">
    <open>Dining</open> <open>Entry</open>
  </Description>
  <Description room="Kitchen">
    <doorway>Hallway</doorway> <open>Entry</open>
    <open>Dining</open> <open>Family</open>
  </Description>
  <Description room="Dining">
    <doorway>Study</doorway> <open>Living</open> <open>Kitchen</open>
  </Description>
  <Description room="Study">
    <doorway>Dining</doorway> <function>Media</function>
  </Description>
  <Description room="Family">
    <open>Study</open> <open>Kitchen</open>
  </Description>
  <Description room="Hallway">
    <doorway id="01">Bedroom</doorway> <doorway id="02">Bedroom</doorway>
    <doorway id="02">Bathroom</doorway> <doorway>Kitchen</doorway>
  </Description>
  <Description room="Bedroom" id="01">
    <open id="01">Bathroom</open> <doorway>Hallway</doorway>
  </Description>
  <Description room="Bathroom" id="01">
    <open id="01">Bedroom</open> <item>Closets</item>
  </Description>
  <Description room="Bathroom" id="02">
    <doorway>Hallway</doorway> <doorway id="03">Bedroom</doorway>
  </Description>
  <Description room="Bedroom" id="02">
    <doorway>Hallway</doorway>
  </Description>
  <Description room="Bedroom" id="03">
    <doorway>Bathroom</doorway>
  </Description>
</RDF>
```

Marisa's completed exercise:



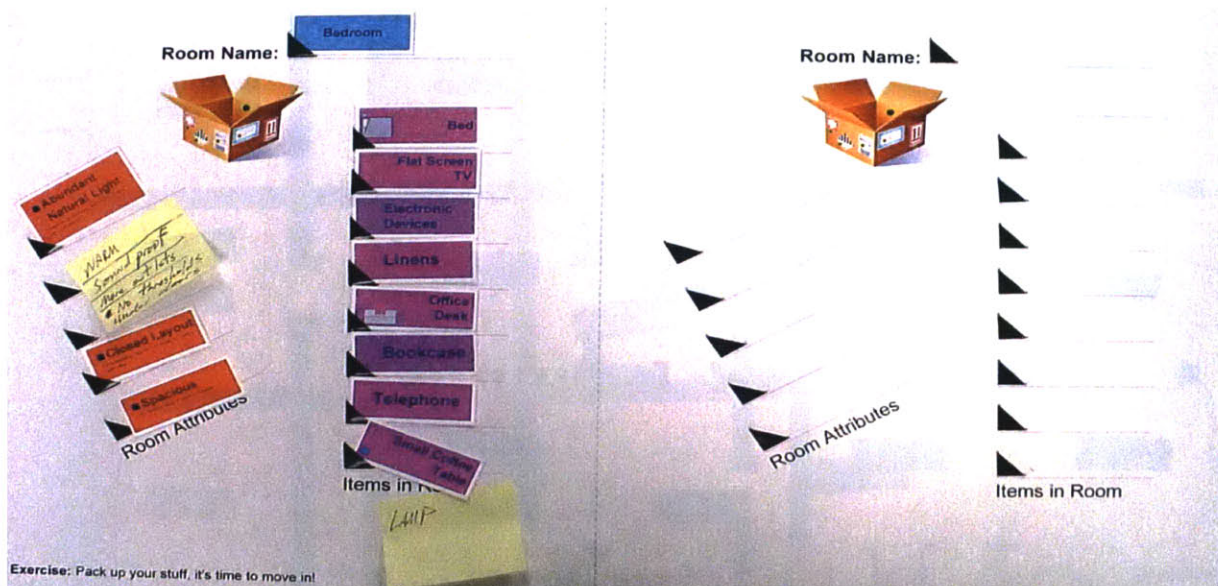
tual Diagramming

A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Role-Playing Metaphor (Packing) 1, by Jeffrey

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Bedroom">
    <finditem>Bed</finditem>
    <finditem>Flat Screen TV</finditem>
    <finditem>Electronic Devices</finditem>
    <finditem>Linens</finditem>
    <finditem>Office Desk</finditem>
    <finditem>Bookcase</finditem>
    <finditem>Telephone</finditem>
    <finditem>Small Coffee Table</finditem>
    <finditem>Lamp</finditem>
    <attribute>Abundant Natural Light</attribute>
    <attribute>Closed Layout</attribute>
    <attribute>Spacious</attribute>
    <attribute>Warm</attribute>
    <attribute>Soundproof</attribute>
    <item count="4">Electrical Outlet</item>
  </Description>
</RDF>
```

Jeffrey's completed exercise:

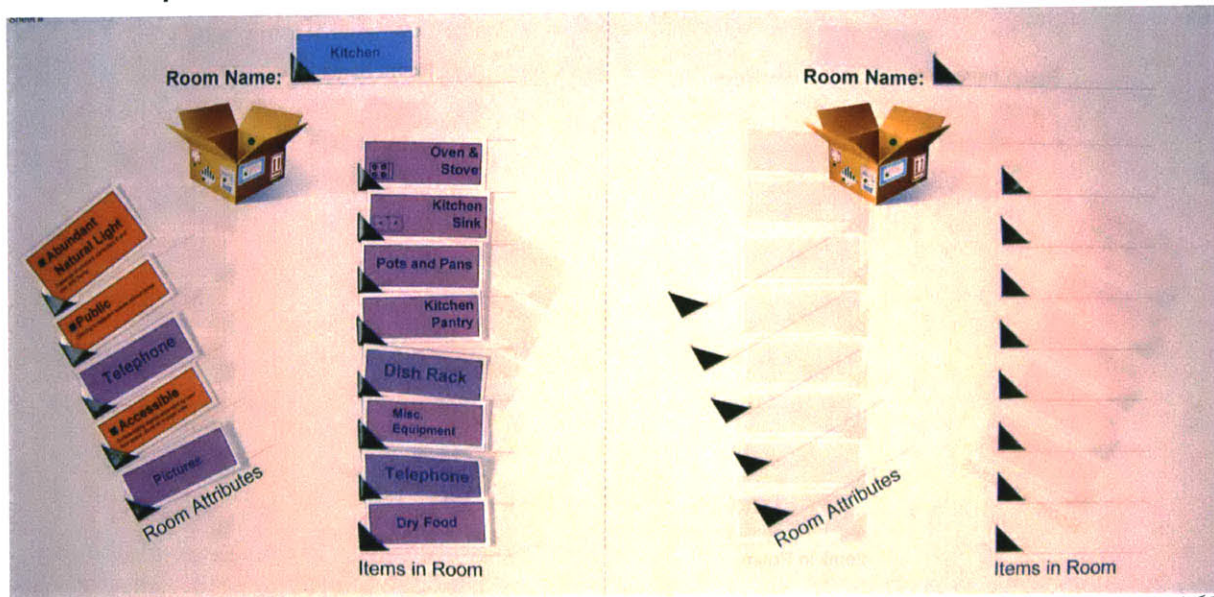


A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Role-Playing Metaphor (Packing) 2, by John

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Kitchen">
    <finditem>Oven and Stove</finditem>
    <finditem>Kitchen Sink</finditem>
    <finditem>Pots and Pans</finditem>
    <finditem>Pantry</finditem>
    <finditem>Dish Rack</finditem>
    <finditem>Misc. Equipment</finditem>
    <finditem>Telephone</finditem>
    <finditem>Dry Food</finditem>
    <attribute>Abundant Natural Light</attribute>
  <attribute>Public</attribute>
  <attribute>Accessible</attribute>
  <item>Telephone</item>
  <item>Pictures</item>
</Description>
</RDF>
```

John's completed exercise:

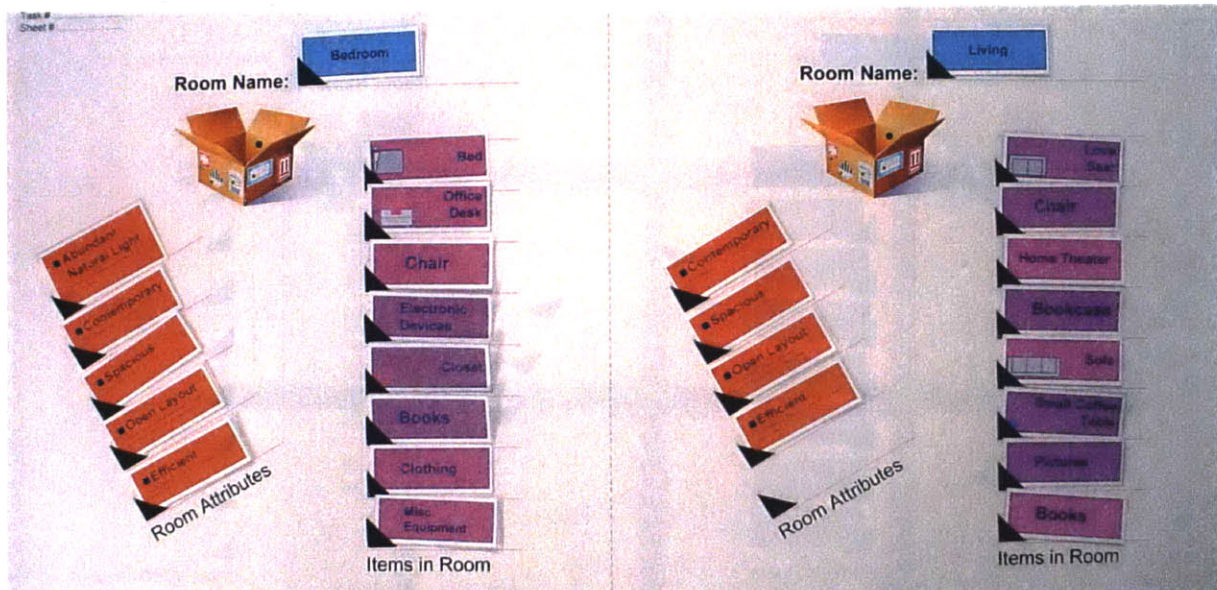


A2.4 The User Exercises Encoded into RDF Queries for Analysis 2

Role-Playing Metaphor (Packing) 3, by Jack

```
<?xml version="1.0"?>
<RDF rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  dc="http://purl.org/dc/elements/1.1/">
  <Description room="Living">
    <finditem>Love Seat</finditem>
    <finditem>Home Theater</finditem>
    <finditem>Chair</finditem>
    <finditem>Bookcase</finditem>
    <finditem>Sofa</finditem>
    <finditem>Small Coffee Table</finditem>
    <finditem>Pictures</finditem>
    <finditem>Books</finditem>
    <attribute>Contemporary</attribute>
    <attribute>Spacious</attribute>
    <attribute>Open Layout</attribute>
    <attribute>Efficient</attribute>
  </Description>
</RDF>
```

Jack's completed exercise (right):



References

1. Baird, George. (2003). *The Space of Appearance*. Cambridge, MA: The MIT Press.
2. Beaudin, Jennifer S. (2003). *From Personal Experience to Design: Externalizing the Homeowner's Needs Assessment Process*. M.S. Thesis Media Arts and Sciences, Massachusetts Institute of Technology.
3. Bergman, Michael K. (2001). "The Deep Web: Surfacing Hidden Value". *The Journal of Electronic Publishing* 7 (1). Retrieved April 15, 2007, from: <http://www.press.umich.edu/jep/07-01/bergman.html>
4. Bush, Vannevar. (1945). "As We May Think", *Atlantic Monthly*, July 1945. Retrieved May 1, 2007, from: <http://www.theatlantic.com/doc/194507/bush>
5. Castells, Manuel. (2004). (Ralph Miliband Memorial Lecture: Mar 14, 2004) *Power and Politics in the Network Society*. Delivered at the London School of Economics and Political Science.
6. Hillyer, Mike (n.d.). "Managing Hierarchical Data in MySQL" Retrieved May 1, 2007, from: <http://dev.mysql.com/tech-resources/articles/hierarchical-data.html>
7. Lieb, Stephen. (1991). "Principles of Adult Learning". *VISION*, Fall 1991. Retrieved March 22, 2007, from: <http://honolulu.hawaii.edu/intranet/committees/FacDevCom/guidebk/teachtip/adults-2.htm>
8. Lynch, Kevin. (1960). *The Image of the City*. Cambridge, MA: Technology Press.
9. Fensel, Dieter, James Hendler, Henry Lieberman, Wolfgang Wahlster. (2003). *Spinning the Semantic Web*. Cambridge, MA: The MIT Press.
10. Fischer, Gerhard, et.al. (1993). "Embedding Computer-Based Critics in the Contexts of Design". *Interchi '93*, April 24th – 29th 1993.
11. Gallagher, Michael P., Alan C. O'Connor, John L. Dettbarn, Jr. and Linda T. Gilday. (2004). *Cost Analysis of Inadequate*

- Interoperability in the US Capital Facilities Industry*. US Department of Commerce: National Institute of Standards and Technology. (Report NIST GCR 04-867).
12. Giarratano, Joseph C., Gary Riley. (2005). *Expert Systems, Principles and Programming*. Boston, MA: Thomson Course Technology.
 13. Habraken, N.J. and M.D. Gross. (1998). "Concepts design games." *Design Studies* 9(3): 150-158.
 14. Harth, Andreas, Stefan Decker. (n.d.). *Optimized Index Structures for Querying RDF from the Web*. Digital Enterprise Research Institute (DERI), University of Galway, Ireland.
 15. Harth, Andreas, Aidan Hogan, Jurgen Umbrich, Stefan Decker. (2007). *YARS2: A Federated Repository For Searching And Querying Graph Structured Data*. Digital Enterprise Research Institute (DERI), University of Galway, Ireland
 16. Hendriks-Jansen, Horst. (1996). *Catching Ourselves in the Act: Situated Activity, Interactive Emergence, Evolution, and Human Thought*. Cambridge, MA: The MIT Press.
 17. index. (n.d.). *The American Heritage® Dictionary of the English Language, Fourth Edition*. Retrieved April 16, 2007, from: <http://dictionary.reference.com/browse/index>
 18. Imparato, Ivo, Diagonal Urbana and Jeff Ruster. (1999). *Participation in Upgrading and Services for the Urban Poor: Lessons from Latin America*. The World Bank.
 19. Jonker, Frederick. (1964). *Indexing Theory, Indexing Methods and Search Devices*. New York, NY: The Scarecrow Press, Inc.
 20. Koile, K. (2001). *The architect's collaborator: toward intelligent tools for conceptual design*. PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
 21. Ma, Xiaoyi. (2002). *A Web-Based User-Oriented Tool for Universal Kitchen Design*. S.M.ArchS Thesis, Department of Architecture, Massachusetts Institute of Technology.
 22. McLeish, T.J. (2003). *A Platform for Consumer Driven Participative Design of Open (Source) Buildings*. M.S. Thesis Media Arts and Sciences, Massachusetts Institute of Technology.
 23. Metcalfe, John. (1976). *Information Retrieval, British & American,*

- 1876–1976. Metuchen, NJ: The Scarecrow Press.
24. Minsky, Marvin. (1985). *The Society of Mind*. New York, NY: Simon & Schuster, Inc.
 25. Minsky, Marvin. (2006). *The Emotion Machine: Commensense Thinking, Artificial Intelligence, and the Future of the Human Mind*. New York, NY: Simon & Schuster, Inc.
 26. Mitchell, William J. (2003). *Me++*. Cambridge, MA: The MIT Press.
 27. Moore, Charles, Gerald Allen, Donlyn Lyndon. (1974). *The Place of Houses*. New York, NY: Holt, Rinehart and Winston.
 28. Mostafa, Javed (2005). "Seeking Better Web Searches" *Scientific American*, February, 2005. Retrieved May 13, 2007, from: <http://www.sciam.com/article.cfm?articleID=0006304A-37F4-11E8-B7F483414B7F0000>
 29. Negroponte, Nicholas. (1975). *Soft Architecture Machines*. Cambridge, MA: The MIT Press.
 30. Papert, Seymour. (1980). *Mindstorms: children, computers, and powerful ideas*. New York, NY: Basic Books.
 31. Perkins, D N. (1981). *The Mind's Best Work*. Cambridge, MA: Harvard University Press.
 32. Piaget, Jean. (1963). *The Origin of Intelligence in Children*. New York, NY: W.W. Norton and Company.
 33. Rao, Satyajit. (1998). *Visual Routines and Attention*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
 34. Reddy, Michael J. (1993). "The conduit metaphor: A case of frame conflict in our language about language", in *Metaphor and Thought*, Andrew Ortony (ed.) Cambridge, England: Cambridge University Press.
 35. Said, Edward W. (2006). *On Late Style: Musica and Literature Against the Grain*. New York, NY: Pantheon Books.
 36. Schon, Donald A. (1983). *The reflective practitioner: how professionals think in action*. New York, NY: Basic Books.
 37. Schuster, J. Mark Davidson, Roger Simmonds, Dennis Frenchman. (1988). *Housing Design & Regional Character: A Primer For New England Towns*. Cambridge, MA: School of Architecture and Planning, Massachusetts Institute of Technology.

38. Said, Edward W. (2006). *On Late Style: Music and Literature against the Grain*. New York, NY: Pantheon Books.
39. Sedgewick, Robert. (1998). *Algorithms in C, Third Edition: Parts 1-4*. Boston, MA: Addison-Wesley Publishing Company, Inc.
40. Salton, Gerard. (1975). *A Theory of Indexing*. Philadelphia, PA: Society for Industrial and Applied Mathematics.
41. Solomon, Michael R. (2003). *Conquering Consumerspace: Marketing Strategies for a Branded World*. New York, NY: AMACOM.
42. Sowa, John F. (1976). "Conceptual graphs for a data base interface". *IBM Journal of Research and Development* 20:4, pp. 336-357.
43. Sowa, John F. (1992). *Semantic Networks*, from "Encyclopedia of Artificial Intelligence", edited by Stuart C. Shapiro. New York, NY: Wiley.
44. Stiny, George. (2006). *Shape*. Cambridge: The MIT Press.
45. Suchman, Lucille Alice. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge, England: Cambridge University Press.
46. Sullivan, Louis H. (1918). *Kindergarten Chats*. New York: Dover Publications, Inc.
47. Tufte, Edward R. (2001). *The Visual Display of Quantitative Information*. Cheshire, CN: Graphics Press.
48. Tufte, Edward R. (1990). *Envisioning Information*. Cheshire, CN: Graphics Press.
49. Ullman, Shimon. *Visual routines*. *Cognition* 18:97-159, 1984.
50. Virvou, M., Katsionis, G., & Manos, K. (2005). "Combining Software Games with Education: Evaluation of its Educational Effectiveness." *Educational Technology & Society*, 8 (2), 54-65.
51. Weinberg, Bella Hass, editor. (1989) "*Indexing: The State of Our Knowledge and the State of Our Ignorance*". Medford, NJ: Learned Information, Inc.
52. Welsh, Matt, Matthias Kalle Dalheimer & Lar Kaufman. (1999). *Running Linux, Third Edition*. Sebastopol, CA: O'Reilly & Associates, Inc.
53. Williams, Reid E. (2003). Training Architectural Computational Critics by Example. M.Eng. Thesis Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

53. Winston, Patrick Henry. (1975). "Learning structural descriptions from examples". in P. H. Winston, ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 157-209.
54. Wittgenstein, Ludwig. (1922). *Tractatus Logico-Philosophicus*. Mineola, NY: Dover Publications.
55. World Wide Web Consortium. (2003). *Scalable Vector Graphics (SVG) 1.1 Specification*. Retrieved May 2, 2007, from <http://www.w3.org/TR/SVG11/>
56. World Wide Web Consortium. (2007). *Semantic Web Activity Statement*. Retrieved May 1, 2007, from <http://www.w3.org/2001/sw/Activity>