Spring 2013

Reconsidering the User

Nathan M. Aleskovsky
Syracuse University

RECONSIDERING THE USER

Nathan M. Aleskovsky
Master of Architecture I Candidate 2013
Syracuse University School of Architecture

Master of Science in Entrepreneurship Candidate 2013
Martin J. Whitman School of Management

Founder & CEO, ShowCode LLC

RECONSIDERING THE USER

My thesis, Reconsidering The User, is a proposal for a digital application that unites the architect and the occupant in the design process of a home by transforming how design criteria are obtained and controlled.

Within the scope of the detached single-family house, my thesis argues that a design process that engages the expertise of both the architect and the occupant has the potential to create a design solution that is more accurately tailored to the preferences of the occupant. This is possible through reconfiguring the information-gathering phase of architectural design. Given my background and current entrepreneurial pursuit,[1] I, along with my advisor, felt that the best way for me to contribute to the field or architecture was not to design a building, but rather how buildings could be built.

My thesis is the culmination and synthesis of several bodies of research within and outside of the field of architecture, which ultimately results my thesis. It is not a proposal for automatic form generation software; I am not attempting to distinguish good designs from bad designs; and I am not suggesting that it is, in any way, superior to the way architects traditionally work.

What I am exploring is (1) an alternative to the way design criteria is gathered from that of a traditional design process and (2) an advancement to the current Do-It-Yourself home design software and floor plan catalogs

In a traditional design process, information collection occurs primarily at the beginning of the design process in an interview. Then the architect uses their expertise and works independently to create a design that the client will routinely review. With the exception of explicitly stated client requirements derived from the interview and subsequent meetings, the architect is free to design as desired (figure 3).

From a floor plan catalog, the future occupant of the home relies on the capitalistic motives of developers. They browse hundreds of floor plans until they find one that is the least objectionable.

Those who choose to design their homes with DIY software typically create homes designs that are infeasible or plagued with problems because they lack the design expertise and are not tailored to the site (figure 4).

The methodology I am proposing eliminates the pre-design interview and implements a communication interface, which (1) facilitates how information is gathered and utilized to influence the design, (2) translates the expertise back and forth between architect and client, and (3) creates an environment in which the client and the architect can simultaneously participate in the design process without compromising the desires of either party (figure 5).

Why is this important? Currently, architects are only directly involved in 2% of single-family home design in the U.S.[2] This is startling considering more than 2/3 of the country lives in detached single-family homes (figure 6).[3]

This means that of the nearly 60 million single-family homes in the U.S., architects were only directly responsible for a little over 1 million. The reasons for this vary a great deal and there is not a definitive conclusion. However, the most prominent causes are (1)Economics and efficiency of detached single family home development,[4] (2) Perception of elitism and exclusivity involved in hiring an architect,[5] and (3) a lack of understanding by the general public about the benefit architects bring to a project beyond aesthetics (figure 7).[6]

I do not, in any way, claim that my thesis is the solution to these concerns, but it recognizes them and attempts to minimize their effects on the design process (figure 8).

In order to accurately present the product of my thesis, it is necessary to quickly elaborate on the areas of research that serve as my proposal's foundation. This diagram (figure 9) is a non-linear, visual representation of my thesis argument that I developed during the course of this project to help me organize and connect the disparate bodies of knowledge.

Before going further, I want to touch briefly on a few concepts that will prove crucial to the understanding of my thesis.

First, User-participatory design, or "Architecture-by-yourself" is concept in which the client and/or the future user of the architecture plays a major role in, and is responsible for, the design decisions.[7]

Historically, in such a practice, the role of the architect is diminished or even eliminated. A clear example of user participatory design is a project called the Flatwriter by Yona Friedman in the 1960's (figure 24).[8] The Flatwriter was a combination of hardware and software in which the future inhabitant of an apartment would select from a series of formal design options in order to create the flat they would eventually live in. In this scenario, the architect is responsible for creating the repertoire of possible solutions for the user from which to select. This project will be detailed later.

Second, I want to distinguish the use of computers in architectural design from that of a computational design process. The traditional use of computers and software for drafting and to aid in the production of drawings and images are simply "more convenient" ways of performing the same process by hand. It is therefore distinct from my reference to a computational process which I define as the method by which an electronic system, constrained by a set of variables.

A proposal for a digital application that unites the architect and the occupant in the design process of a home by transforming how design criteria are obtained and controlled.

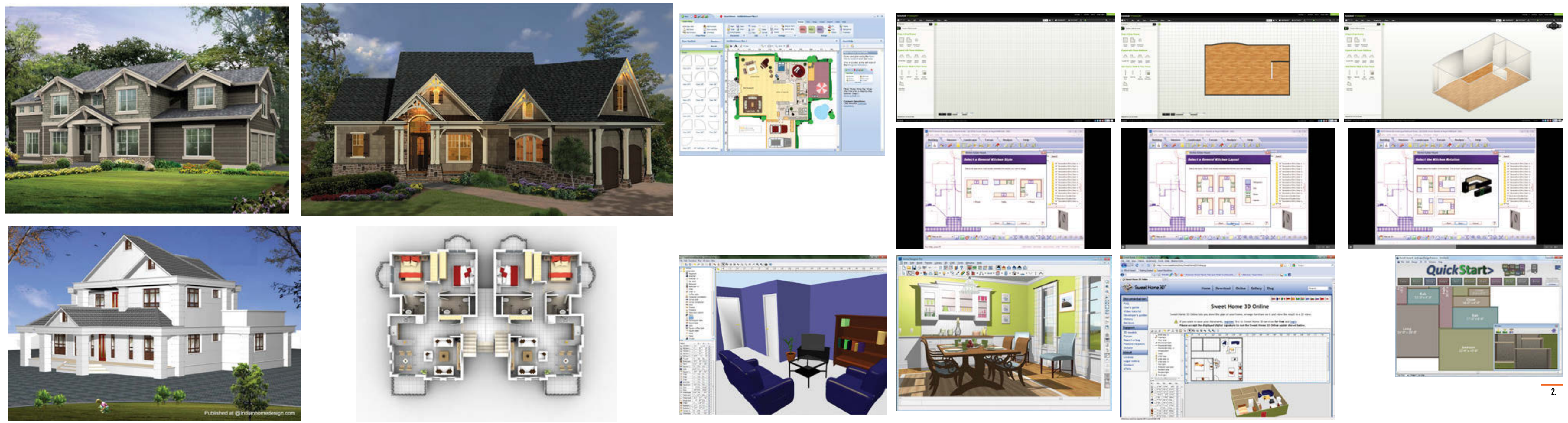
Within the scope of the detached single family house,
a constraint-based design process
that engages the expertise of both the architect and the occupant
has the potential to create a design solution that is more
accurately tailored to the
preferences of the occupant.

Figure 1. Diagram of the entire thesis scope.

an alternative to the way design criteria is currently gathered

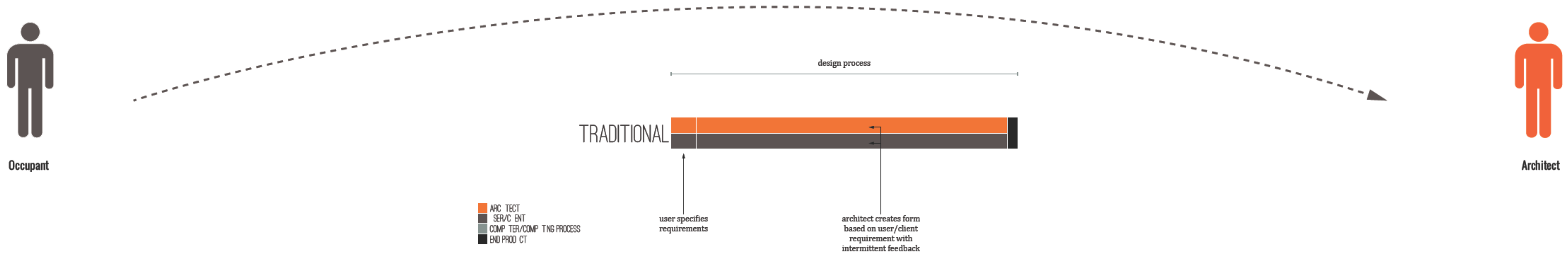
- and -

an advancement to the current DIY home design software and floor plan catalogs



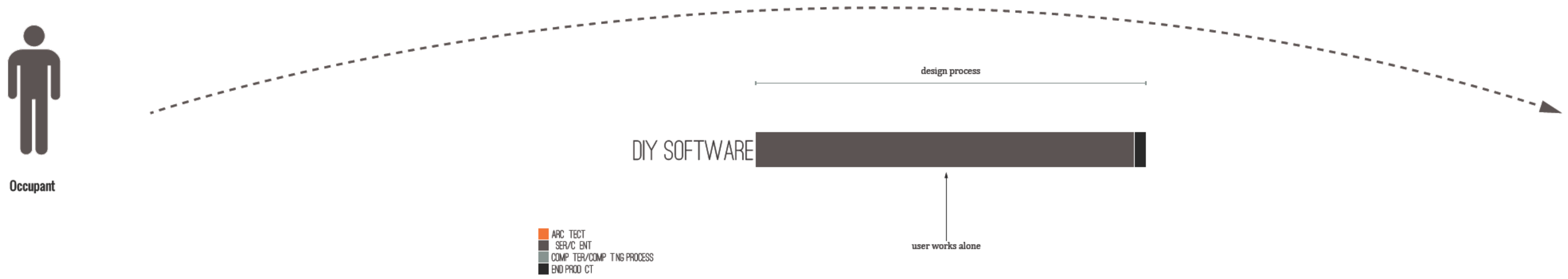
2.

Figure 2. Examples of the software and 3D models used to of current at-home DIY software packages as well as examples of catalog home designs.



3.

Figure 3. Diagram of the interaction of architect, client, and computation in traditional architecture practice



4.

Figure 4. Diagram of the interaction of architect, client, and computation with DIY software and developer catalogs

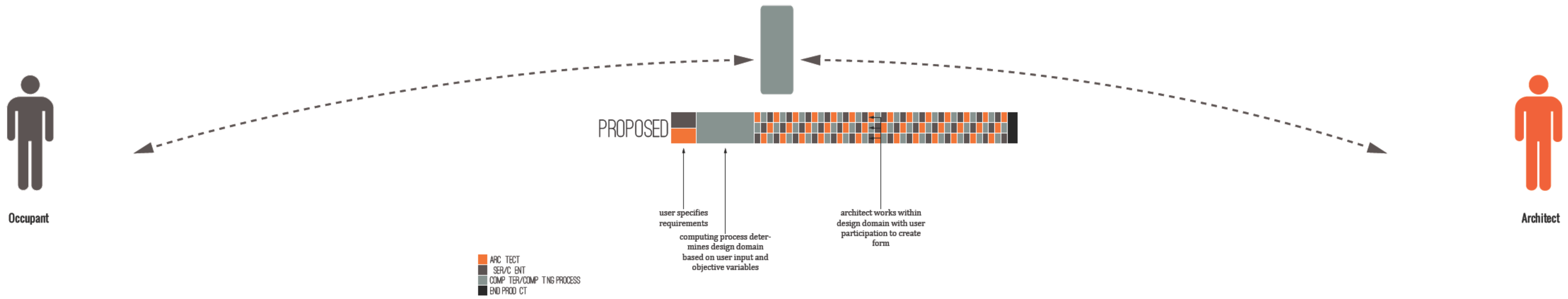
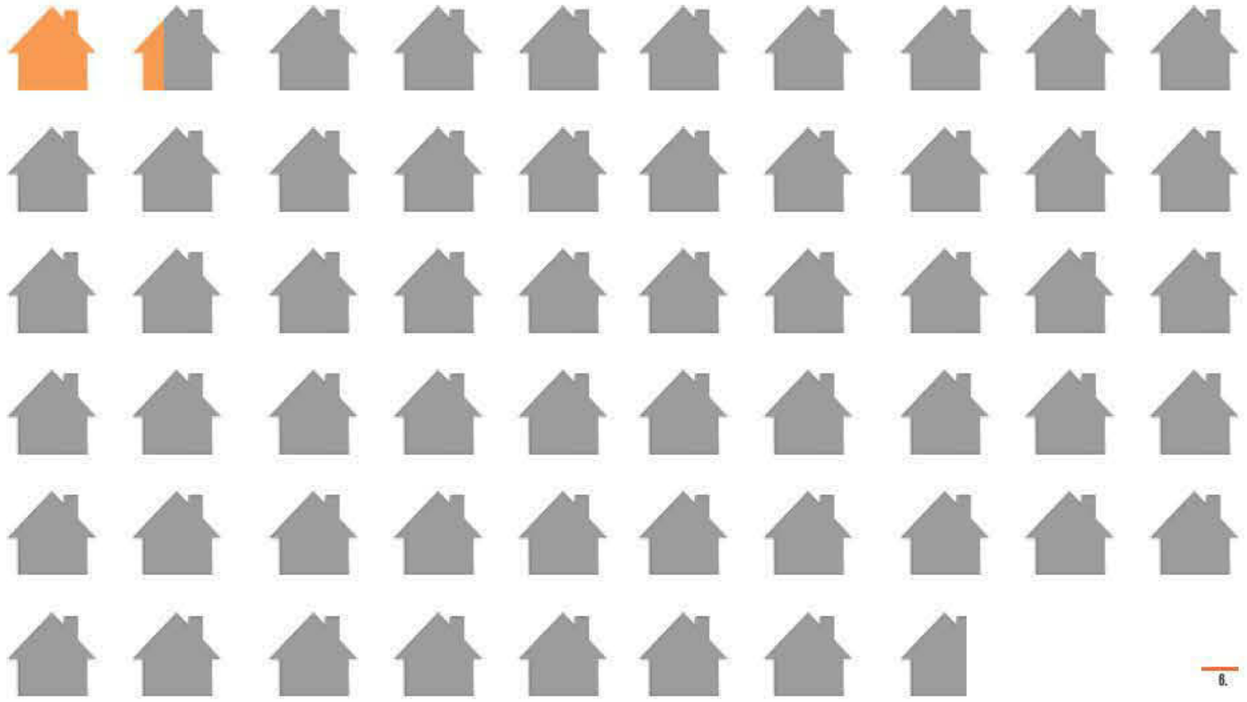


Figure 5. Diagram of the proposed interaction of architect, client, and computation.



Economics and efficiency of detached single family home development

LaBarre Suzanne 2008 Truth in Numbers | Metropolis Magazine" Metropolis Magazine October 15 http://www metropolismag com/story/20081015/truth-in-numbers
U S Census Bureau Demographic Internet Staff 2013 "Residential Finance Survey Main" Accessed April 24 http://www census gov/housing/rfs/

Perception of elitism and exclusivity involved in hiring an architect

LaBarre Suzanne 2008 Truth in Numbers | Metropolis Magazine" Metropolis Magazine October 15 http://www metropolismag com/story/20081015/truth-in-numbers

A lack of understanding by the public about the benefit architects bring to a project beyond aesthetics.

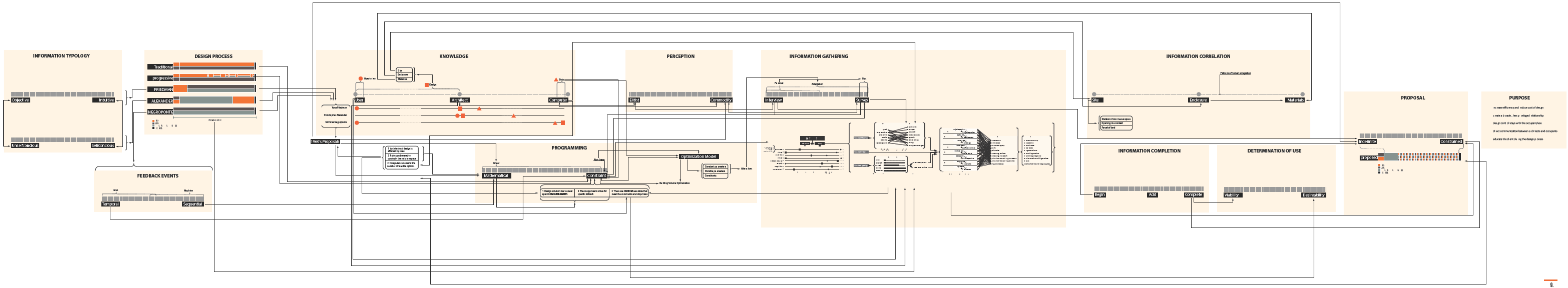
Thompson Max 2012 "It s True: People Don t Know What Architects Do | News | Architects Journal" Architects Journal July 19 http://www architectsjournal co uk/news/daily-news/its-true-people-dont-know-what-architects-do/8633240 article

- increase efficiency and reduce cost of design
- create a broader, less privelaged relationship
- design control stays with the occupant/user
- direct communication between architects and occupants
- educate the client during the design process

Figure 6. "Yona Friedman". from http://www.ina.fr//images_v2/fresques/imagettes/europe/jpegVisionneuse/Europe00061.jpg

Figure 7. LaBarre, Suzanne. 2008. "Truth in Numbers | Metropolis Magazine." Metropolis Magazine. October 15. http://www.metropolismag.com/story/20081015/truth-in-numbers.; U. S. Census Bureau, Demographic Internet Staff. 2013. "Residential Finance Survey Main." Accessed April 24. http://www.census.gov/housing/rfs/.; Thompson, Max. 2012. "It's True: People Don't Know What Architects Do | News | Architects Journal." Architects Journal. July 19. http://www.architectsjournal.co.uk/news/daily-news/its-true-people-dont-know-what-architects-do/8633240.article.

Figure 8. Proposed solutions to the recognition of industry problems



9.

Figure 9. "Yona Friedman". from http://www.ina.fr//images_v2/fresques/imagettes/europe/jpegVisionneuse/Europe00061.jpg

HISTORICAL CONTEXT

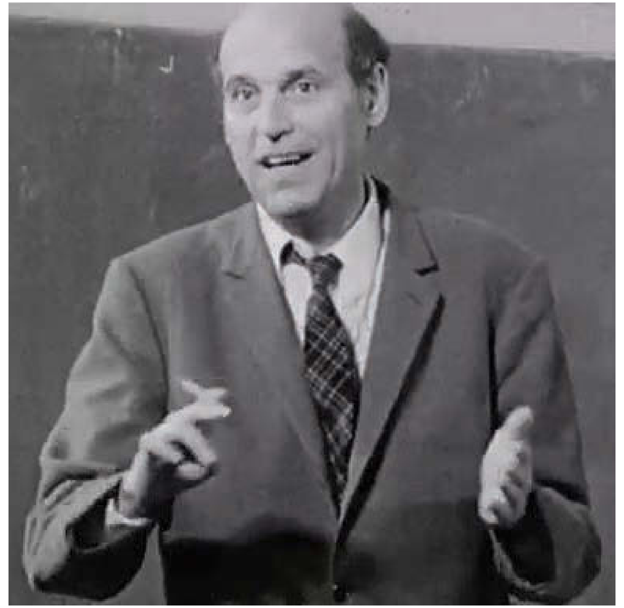
In the 1960s and 1970s, many individuals did research into user-participatory design and architectural computing. Among those whose work is most noteworthy, as evidenced by their continued relevance in contemporary discussions, is Nicholas Negroponte, Christopher Alexander, and Yona Friedman.

Each of these architects developed methodologies that utilize technology and computing as the armature of a design process and, to some extent, propose a design system that favors analytical and/or logical thinking over intuitive thinking. Each methodology creates a technological platform from which to work and results in a sample of differentiated projects that are each realized by implementing varying degrees of end-user participatory design. As one who – beyond the realm of this thesis – is studying, participating in, and actively developing a system in which computing and technology play a significant role in the design process, I am intrigued by their distinctive conclusions, specifically as they relate to the roles of the end user and the architect.

The technology- and computing-based design processes developed by these three authors are valid and cogent propositions that have withstood the evolution of technology and computing over the past five decades and are as valid now as they were decades earlier.

I will begin this argument by providing an overview of each author and their respective theories as derived from their written work(s) relating to the use of technology and computing in the design process. The overview will serve two purposes. First, it will provide a background for readers who are not closely familiar with the work of these authors. Second, it will establish some boundaries regarding the scope of my thesis. Each of these architects over the course of their lives has been associated with larger arguments regarding, among others, phenomenology and the importance of place. However, the scope of this thesis will be limited to the application of their respective theories and systems regarding the use of technology and computing in the design process. Through this structure, I will pay specific attention to the proposed design process that resulted from the theoretical argument as well as the realized projects.

I will begin with and spend more time discussing Yona Friedman's work in order to provide clarity to the computational architectural design process, the backbone of which can be applied to the subsequent theories.



YONA FRIEDMAN

10.

Figure 10. "Yona Friedman". from http://www.ina.fr//images_v2/fresques/imagettes/europe/jpegVisionneuse/Europe00061.jpg

Figure 11. Redrawn from Friedman, Yona. 1980. *Toward a Scientific Architecture.* Cambridge, Mass.: MIT Press.

Figure 12. Ibid.

In Toward a Scientific Architecture, "Friedman's main objective is to "democratize" design, to free the user from the "patronage" of the architect, to enable "non experts" to make their own designs, as they are the ones who better know their needs and desires and, most importantly, bear the risk of failure."[9]

Friedman argues that in the past, architecture consisted of a "simple chain of operations"[10] in which the architect worked directly with the client and future user. In its most basic form, the future user makes decisions directly about the finished product.

However, he argues that as buildings became more complex, the architect became involved in the process. The future user conveyed his specific needs for his building directly to the architect who, in turn, translated the future user's needs into the design of the finished product. In this arrangement, the architect essentially does not exist in the decision making process. The architect was the middleman between the client and the builder but "all the decisions had been made exclusively by the client."[11] What has changed in the present (1960's) is that the architect now works for thousands of future users and it is thus impossible for the architect to consider all of the needs and requirements of every future user when designing the building. He argues that the industry is left with two solutions:

"1. Supply a large enough number of architects…so that each of them can devote himself to a very few clients.
2. Reduce the period of time spent gathering information (between the client's visit and the construction of the hardware)"[12]

Given that it would require an unbelievable number of architects to make option 1 feasible, the industry has chosen option 2. The result is that instead of designing for each individual user, architects now design for the specific needs of the average future user.

The problem with this approach, he states, that the average user does not exist. To express this in an extremely simplified manner: the architect has gone from designing for one user, to designing for thousands of users, to designing for no user.

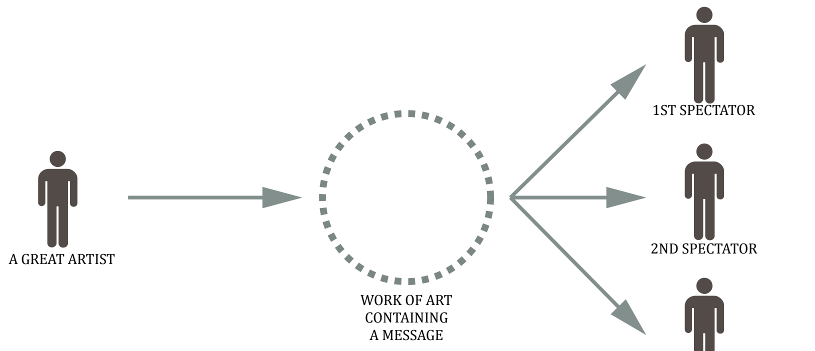
Notice that in figures 6 and 7, there is a bottleneck where the information is being received by the architect. Friedman seeks to eliminate the bottleneck by implementing a feedback loop (figure 8). He claims that constructing this new process will, "eliminate information short circuits and therefore unreliability from the message on arrival",[13] in other words, 'noise'" as seen in figure 8.

Friedman states, "The act of deciding also implies that the one who makes the decisions is the one who takes the risks. Any system that does not give the right of choice to those who must bear the consequences of a bad choice is an immoral system. However, that is exactly the way that architects and planners work. They make the decisions and the users take the risks."[14]

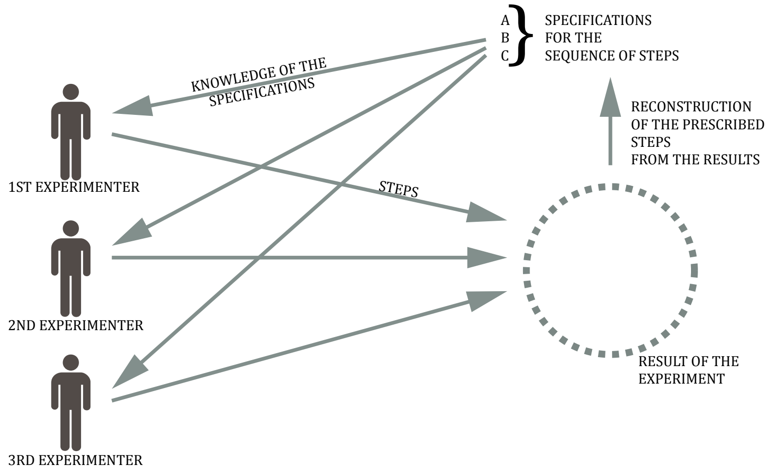
Through this process, Friedman recognizes that the future user must thoroughly understand the risks involved in making design decisions, stating that it is, "immoral and dangerous to leave choices to people who have not been properly informed about the consequences of their decisions".[15] Freidman argues that the role of the architect should be to construct the repertoire that the occupants use, instead of designing the spaces in which they occupy.

This thinking paved the way for Friedman's Flatwriter a hardware and software solution that allows the end user or occupant to design their housing unit to their exact specifications based on how often they used the space and the positioning of programmatic spaces in relation to one another.

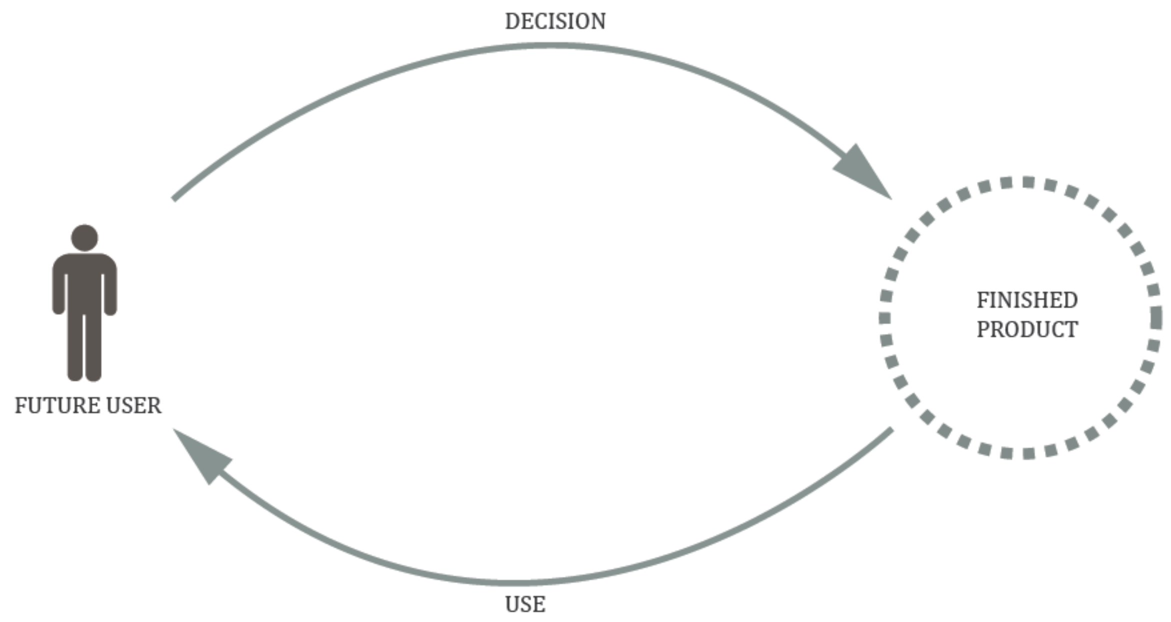
To accomplish this he diagrams possibilities of connections between exterior spaces to interior spaces and interior spaces to one another. The following are a series of diagrams originally drawn by Friedman that I have redrawn here for clarity (figures 10, 11, 12, and 13).

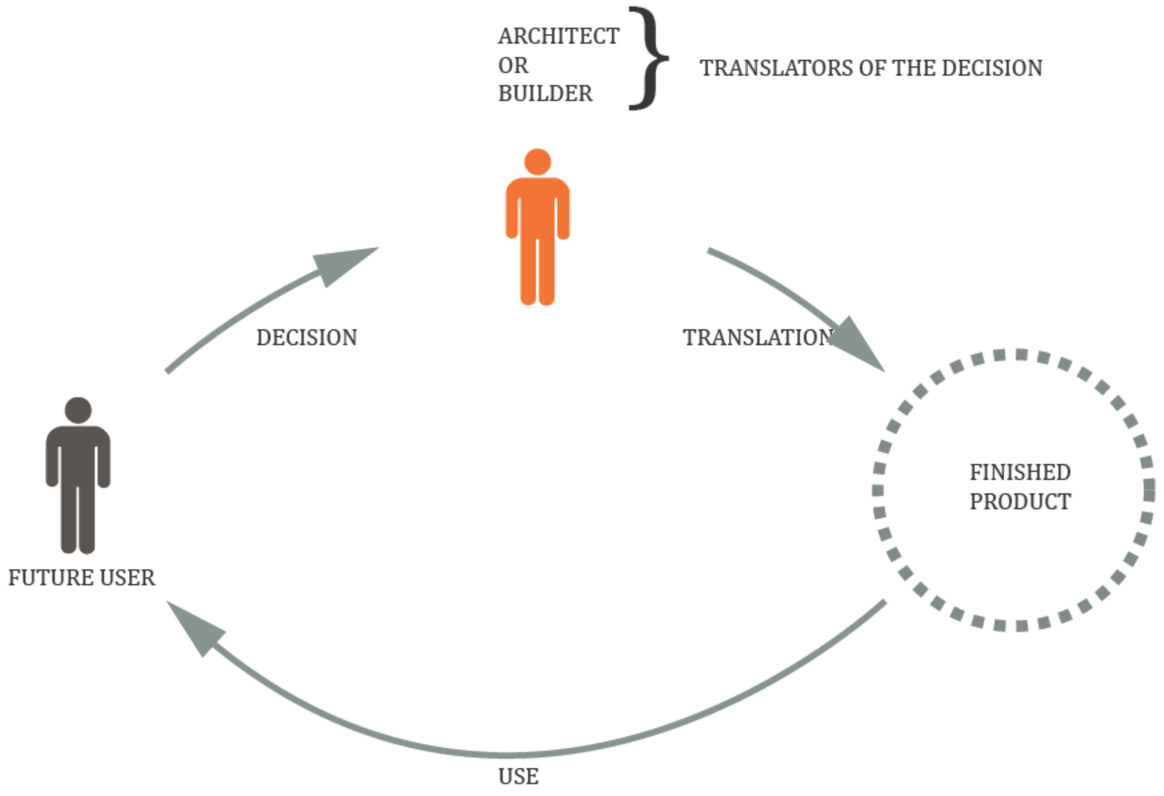


11.



12.





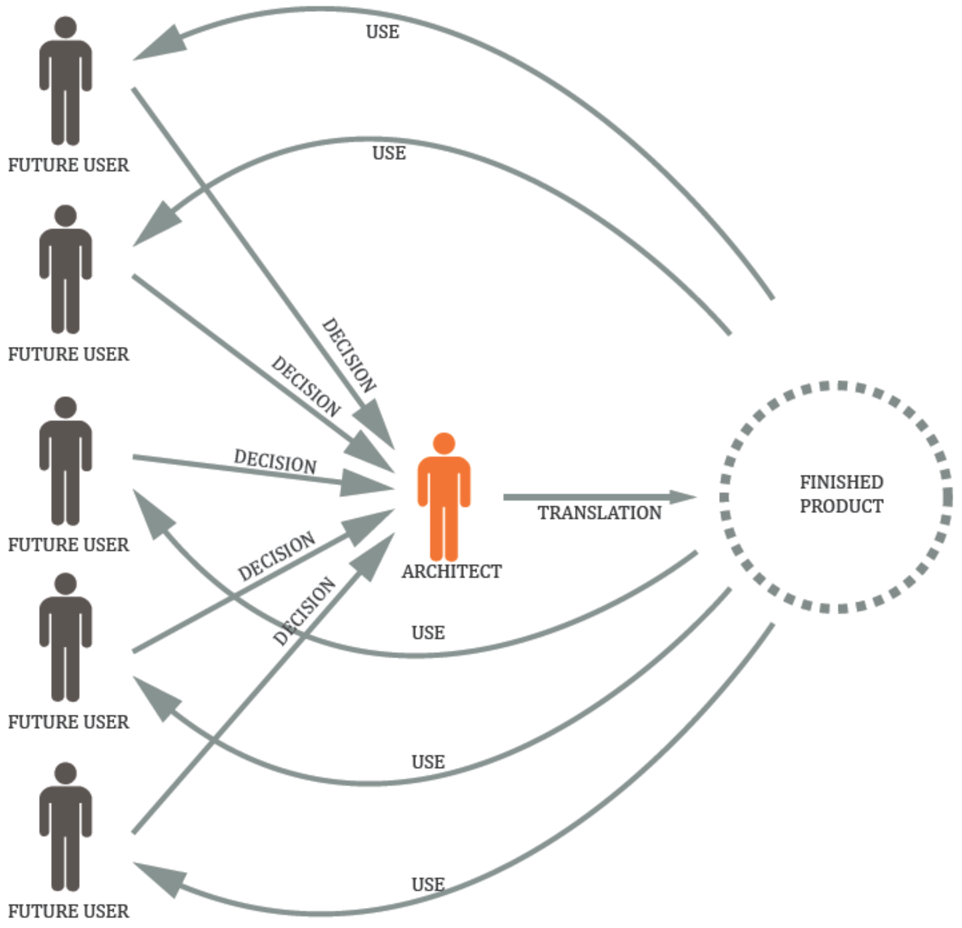
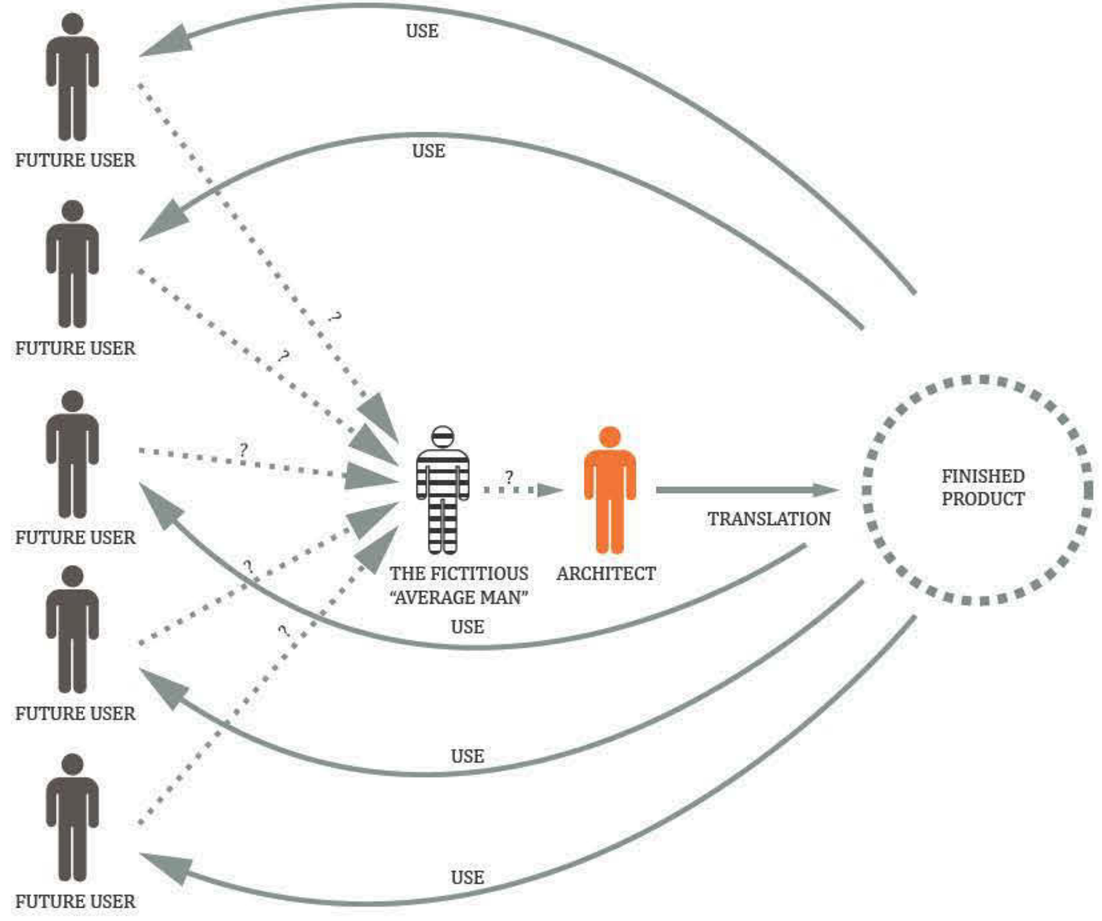
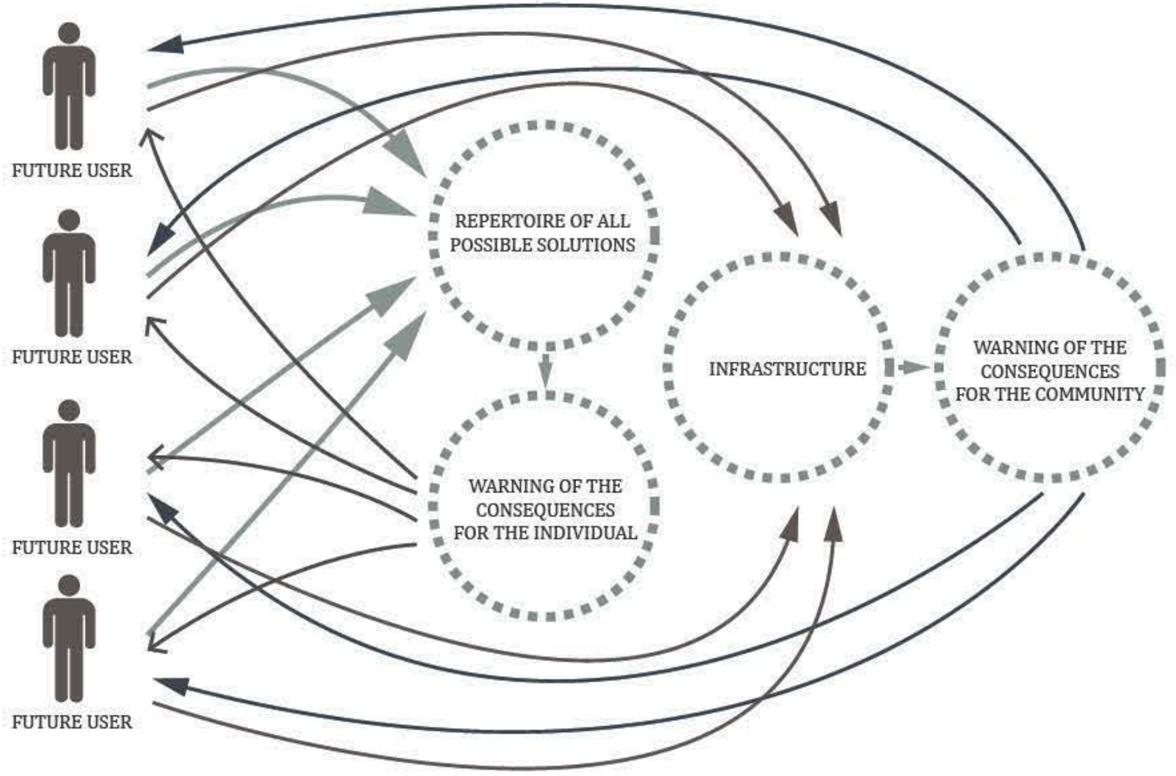


Figure 13. Redrawn from Friedman, Yona. 1980. *Toward a Scientific Architecture*. Cambridge, Mass.: MIT Press.

Figure 14. Ibid.

Figure 15. Ibid.





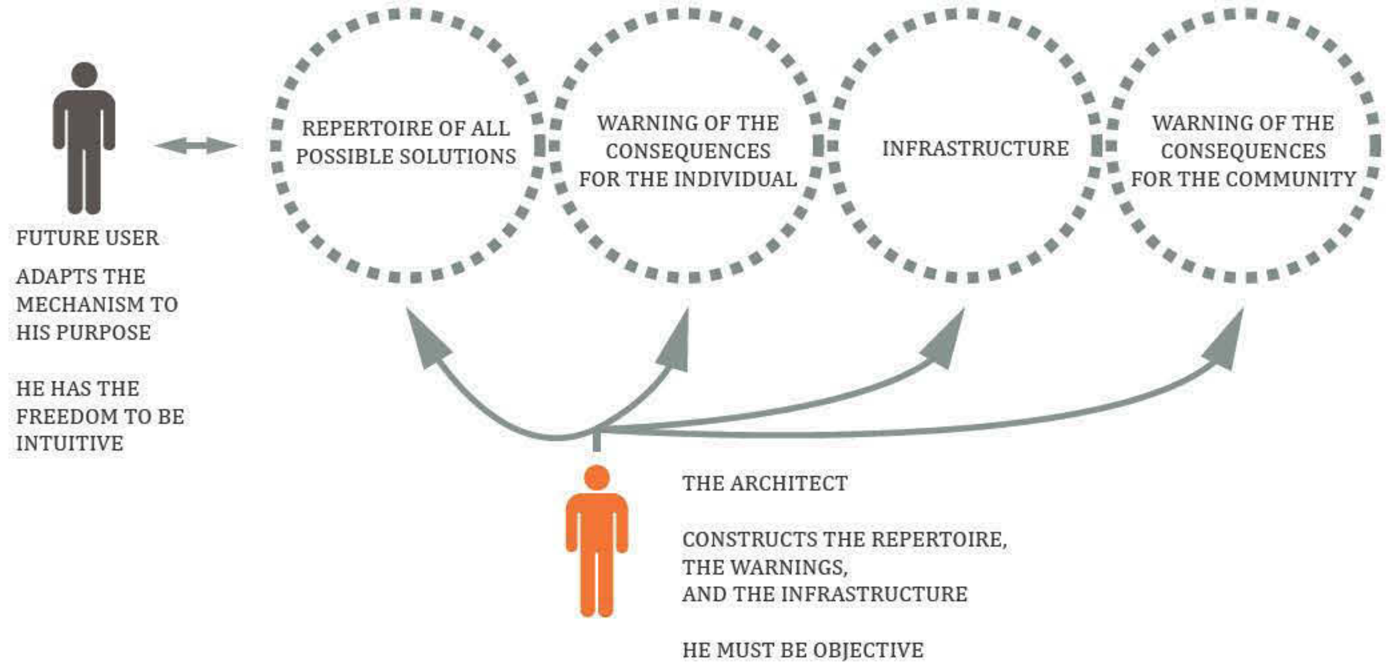
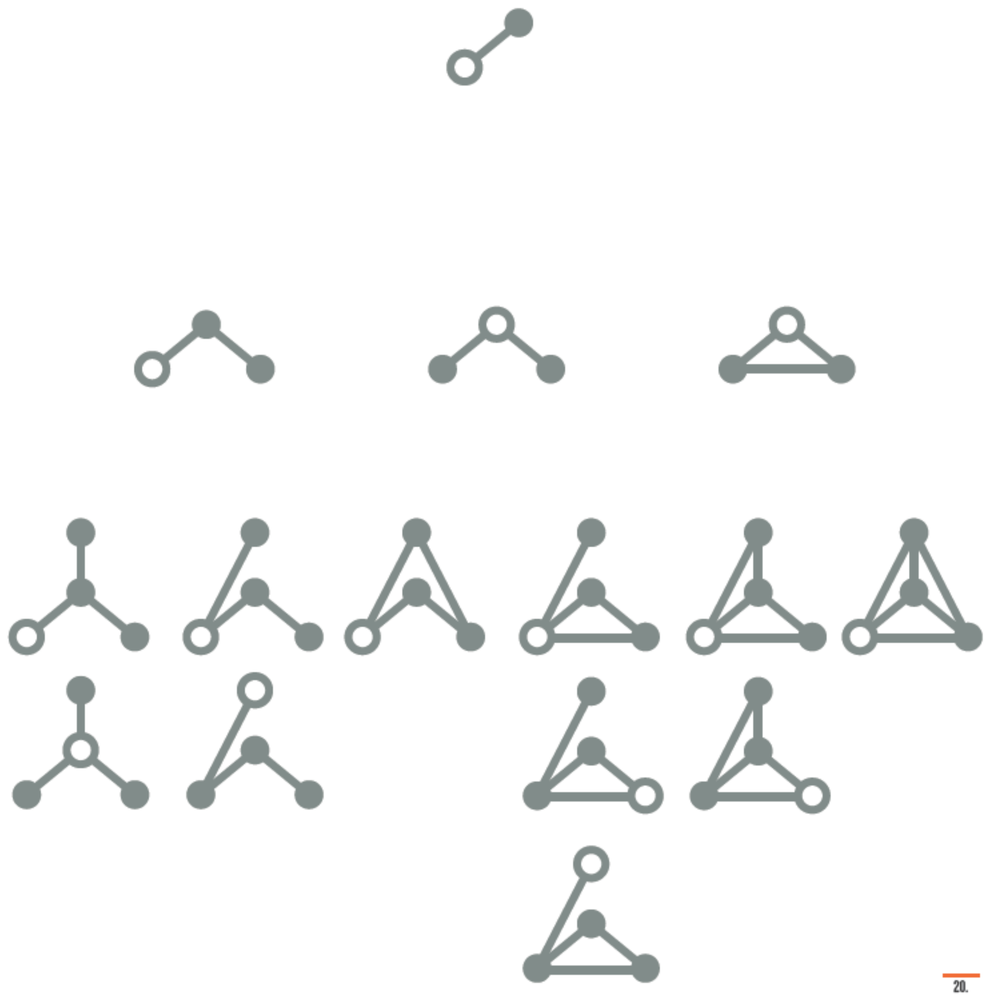


Figure 16. Redrawn from Friedman, Yona. 1980. *Toward a Scientific Architecture*. Cambridge, Mass.: MIT Press.

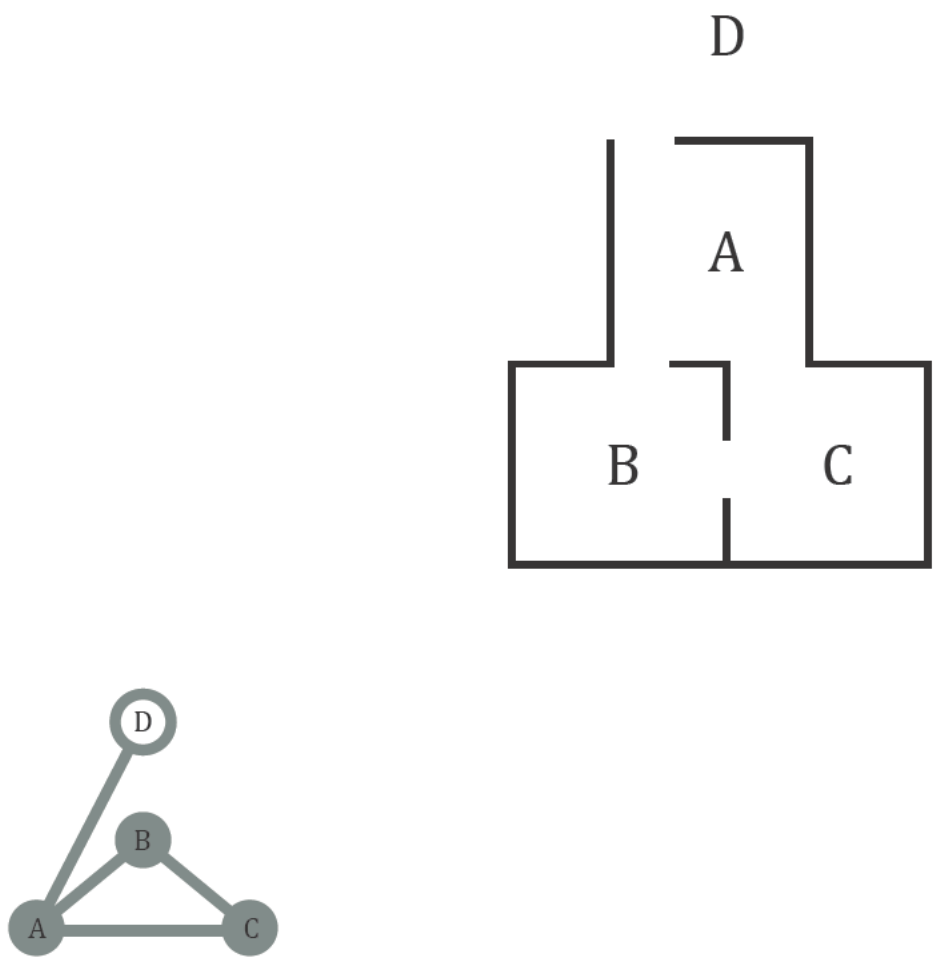
Figure 17. Ibid.

Figure 18. Ibid.

19.



20.

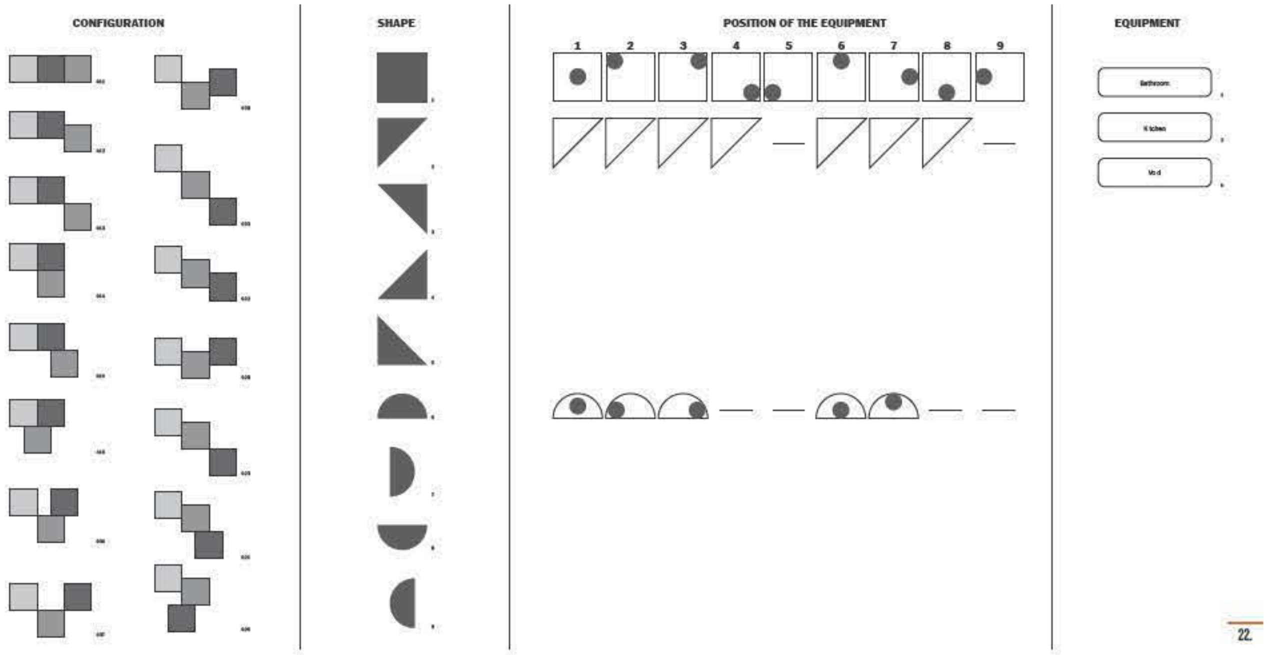


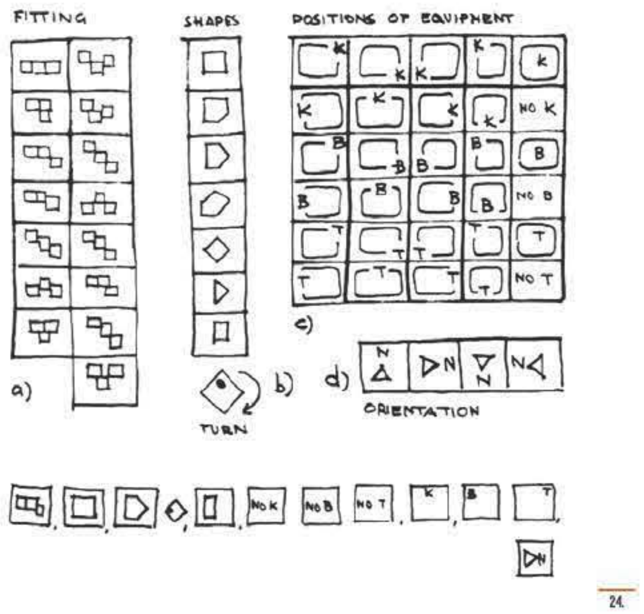
21.

Figure 19. Redrawn from Friedman, Yona. 1980. *Toward a Scientific Architecture*. Cambridge, Mass.: MIT Press.

Figure 20. Ibid.

Figure 21. Ibid.





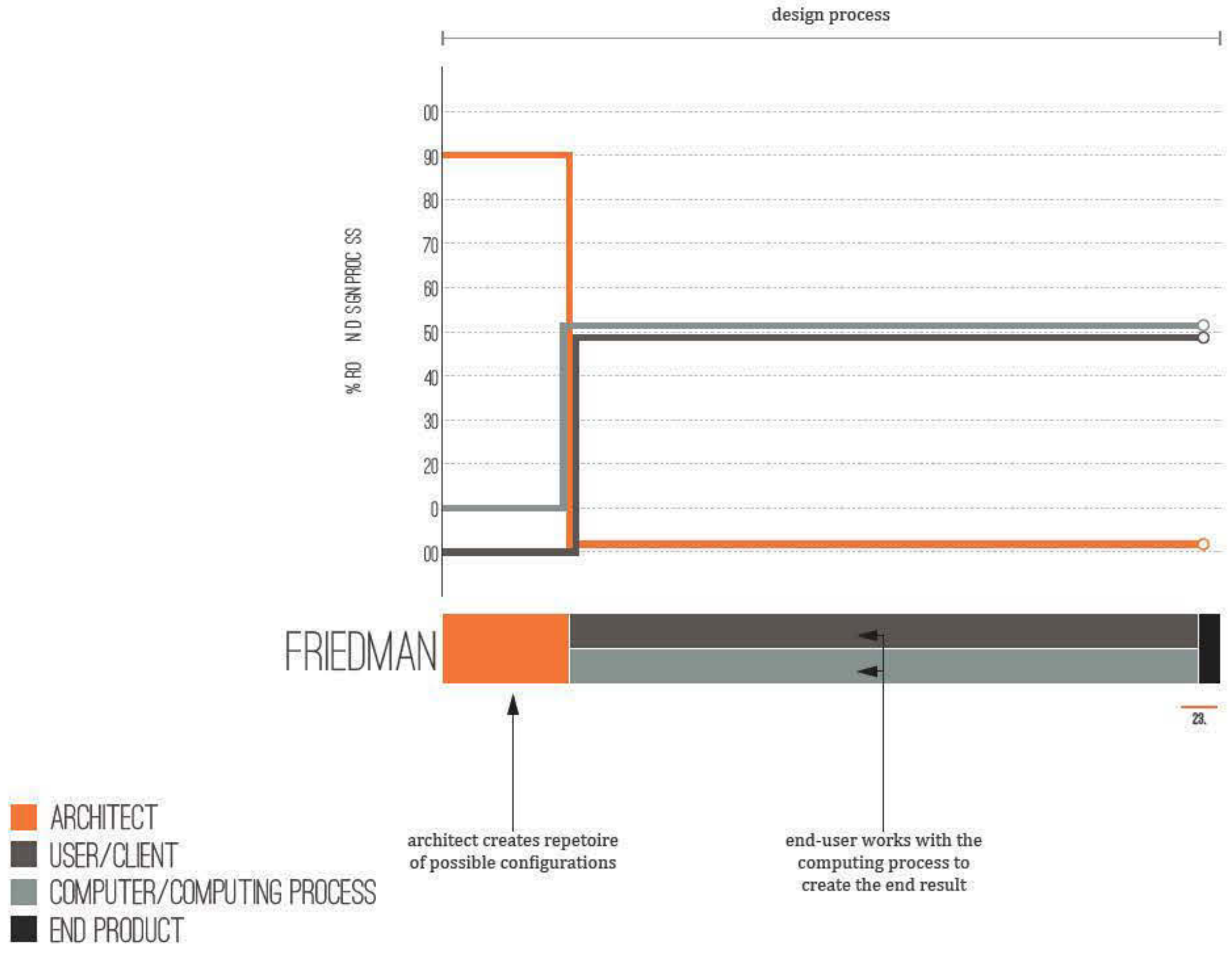


Figure 22. Redrawn from Friedman, Yona. 1980. *Toward a Scientific Architecture*. Cambridge, Mass.: MIT Press.

Figure 23. Diagram illustrating the roles of the architect, computer, user, throughout the design process using Friedman's Flatwriter.

CHRISTOPHER ALEXANDER

25.

Figure 24. "Christopher Alexander". from http://blog.buildllc.com/wp-content/uploads/2010/10/Christopher-Alexander.jpg

Figure 25. Redrawn from Alexander, Christopher. 1964. *Notes on the Synthesis of Form*. London: Oxford University Press.

Figure 26. Ibid.

Figure 27. Ibid.

Like Friedman, Alexander also proposes a systematic process to conceive form; relying on the result that provides the best fit. In a basic design problem, the designer must meet requirements to find an appropriate solution. There are interactions between these requirements, which make them hard to meet. A simple problem, he argues "falls easily within a single man's intuition, but a more complex problem cannot be solved by intuition alone."[16]

Figure 16 is Alexander's diagram of the connections involved in the form making process. The variables, represented by the points, are interconnected to one another and are therefore not only dependent, but also interdependent - Preventing the variables from adjustment.

Therefore, there will always be subsystems (figure 17) that allow for independent adjustment. Form making he argues, "is the action of a series of subsystems, all interlinked, yet sufficiently free of one another to adjust independently in a feasible amount of time."[17]

In defining the context and requirements for a design problem, Alexander recommends that designers should "see the process of achieving a good fit between two entities as a negative process of neutralizing the incongruities, or irritants, or forces which cause misfit".[10] In other words, a "good fit [is the] absence of certain negative qualities."[18]

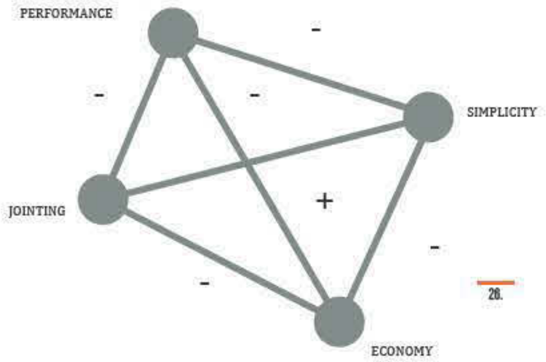
Why? A designer could list all the requirements of a design, but that list is potentially endless. To remedy this dilemma, we need to develop a field description that is comprised of a finite set of variables. "If we think of the requirements from a negative point of view, as potential misfits, there is a simple way of picking a finite set. This is because it is through misfit that the problem originally brings itself to our attention."[19]

Even after this, he states that while compressed, the list is potentially enormous. To distill this information, the human mind classifies and categorizes the variables in order to limit the number of distinct concepts.[20]

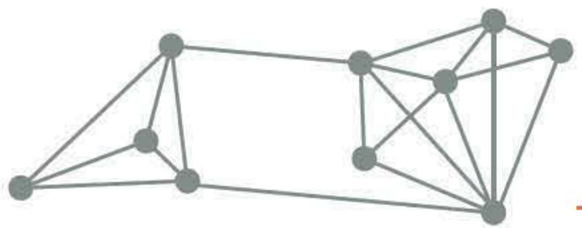
As an example, Alexander discusses the design of a teakettle. If one must design a kettle there are an infinite number of variables that the designer must deal with. In his book, he fills a page of variables to illustrate the complexity. Through a system of classification, we create a hierarchy similar to Alexander's diagram here. Each of the nodes on this diagram can be considered a "set" of variables.

"The starting point of analysis is the requirement. The product of analysis is the program, which is a tree of sets of requirements (figure 18). The starting point of synthesis is the diagram. The product of synthesis is the realization of the problem, which is a tree of diagrams. The program is made by decomposing a set of requirements into successively smaller subsets."[21] In a subsequent article, "A City is not a Tree" Alexander softens the rigor of his tree analysis.

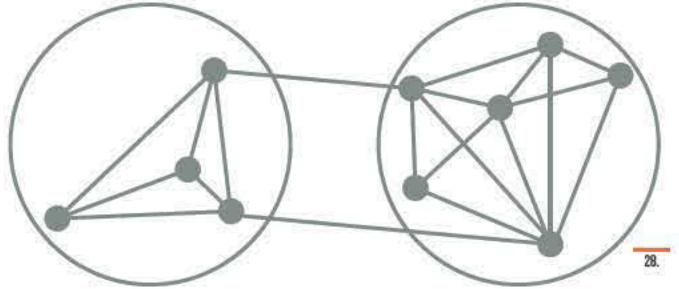
Because this thesis is not a critique on his theory, an analysis of that text is outside the scope of this thesis. I bring up this Alexander's work because, despite the era in which it was written, his theories lay the foundation for a methodology that has only recently been utilized create architectural form.



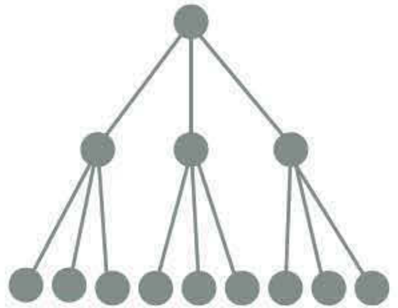
26.



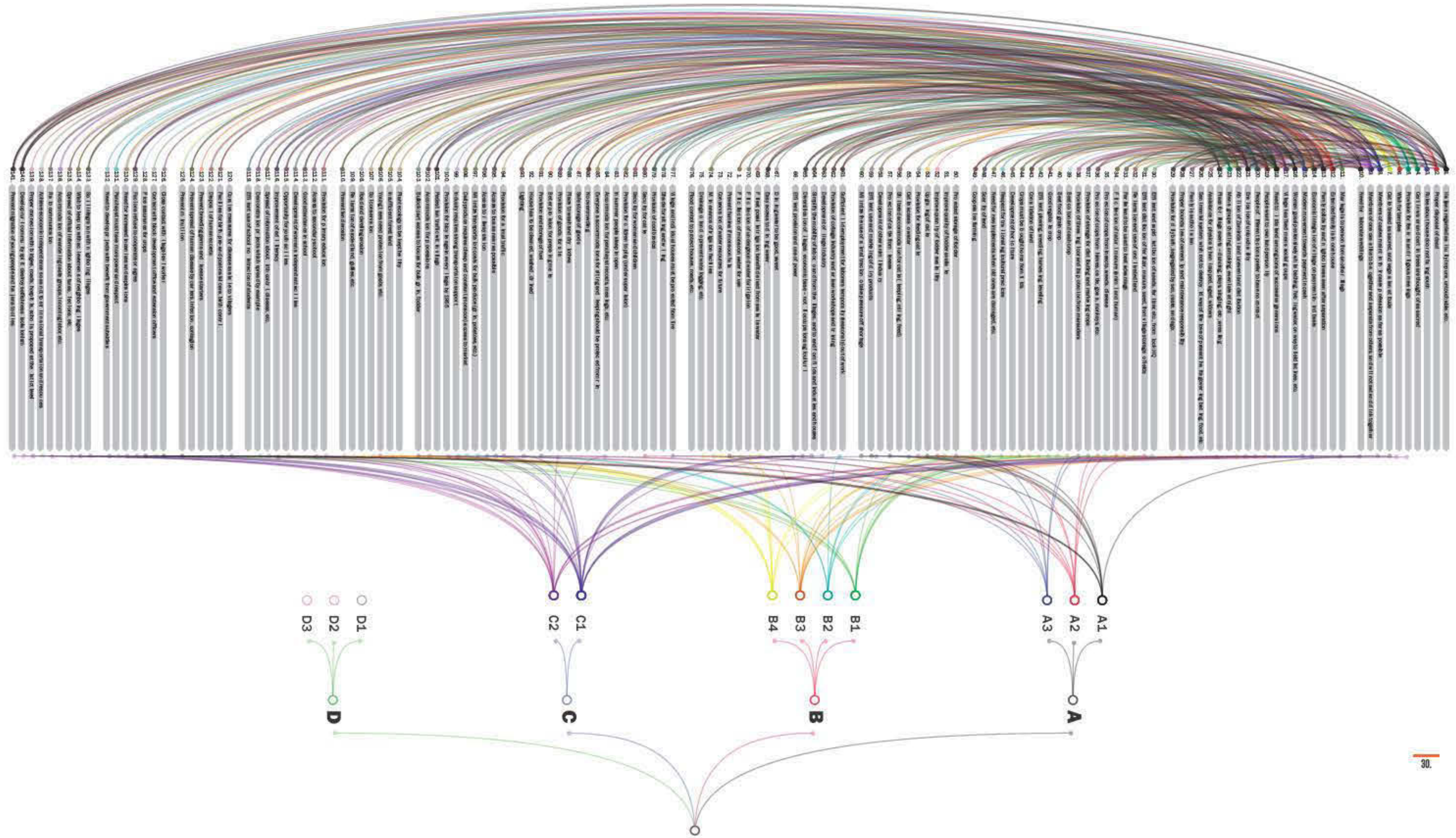
27.



28.



29.



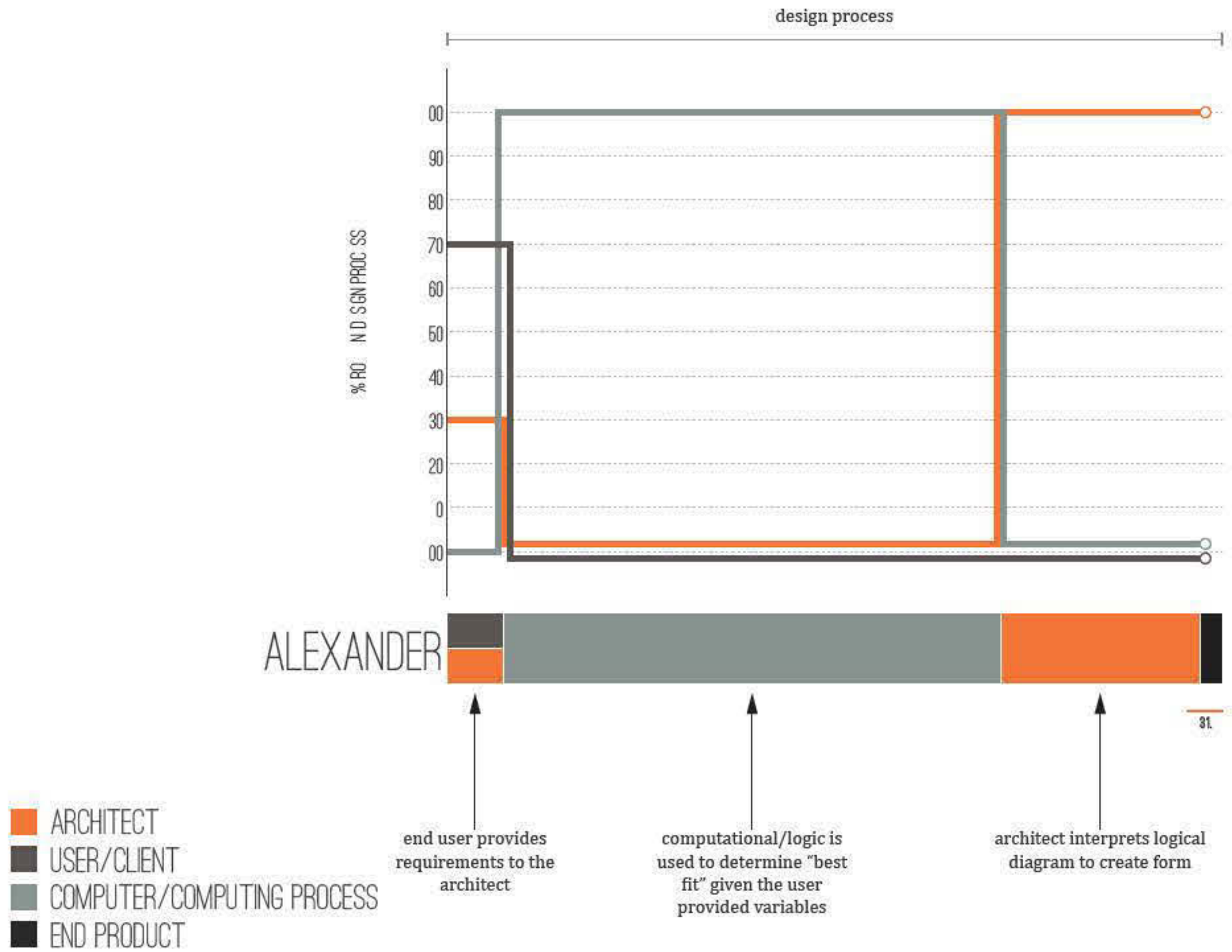
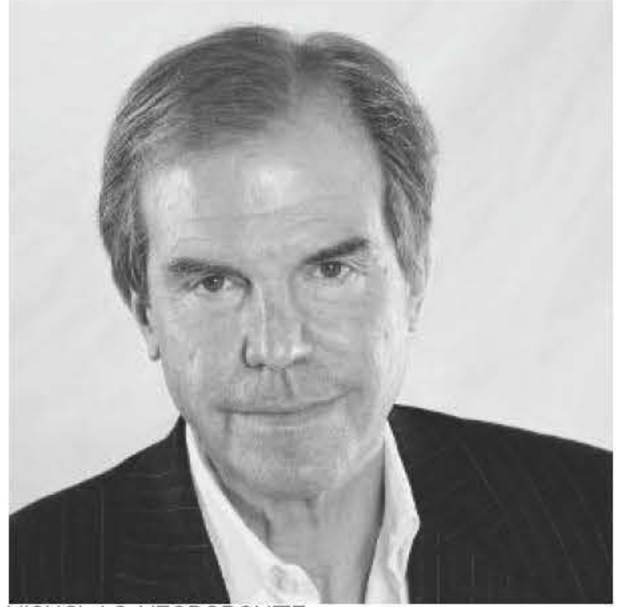


Figure 28. Visualization of Christopher Alexander's logic-based design methodology for the design of a village. Aleskovsky

Figure 29. Diagram illustrating the roles of the architect, computer, user, throughout the design process using Alexander's theory.



NICHOLAS NEGROPONTE

32.

Figure 30. "Nicholas Negroponte" from http://one.laptop.org/sites/default/files/imagecache/thumb_large/NN-300.jpg

Figure 31.Redrawn from Negroponte, Nicholas. 1972. *The Architecture Machine*. Cambridge, Mass.: MIT Press.

Figure 32. Redrawn from Negroponte, Nicholas. 1972. *The Architecture Machine*. Cambridge, Mass.: MIT Press.

Figure 33. Redrawn from Negroponte, Nicholas. 1972. *The Architecture Machine*. Cambridge, Mass.: MIT Press.

In referring to Negroponte, I will be focusing my analysis to the work conducted by URBAN5. Urban5's original goal was to "study the desirability and feasibility of conversing with a machine about an environmental design project...using the computer as an objective mirror of the user's own design criteria and form decisions; reflecting responses from a larger information base than the user's personal experience ... the object was to develop a system that could monitor design procedures, in effect, be an urban design clerk."[22]

The program process for URBAN5 contained feedback loops (which turned out to be consistent with Friedman's theory) but was ultimately a linear process. I will discuss why this is a problem in more detail a little later, but essentially, the program found conflicts because it was searching for a single answer to a particular criterion.

Negroponte identified two types of events: temporal and sequential. "Together they constitute part of a process", Negroponte states, "A sequential response of one protagonist is generated by the previous event in the dialogue, usually on behalf of the other. A sequential event is a reply."[22] In order for the system to work, "a sequential episode assumes the reply of one system [(the computer)] and the attention of the other system".[23] Temporal events on the other hand are unsolicited interruptions.

In figure 34, originally drawn by Negroponte in his book *The Architecture Machine*, the computer is presented with criteria that it attempts to mitigate in search of an appropriate form. Because URBAN5 was intended to provide an interaction with the architect user, the computer would propose a form and ask the user if it was 'good' or 'not good'. If it were good, the computer would continue to develop the model.

This first portion of the diagram represents a sequential event in which the computer and the user are interacting. If the form is not good, the computer makes the decision to test the criteria again, and elicit a response from the user, continuing the sequential event. If an appropriate form is not found after repeating this process, the computer requires the user to change the criterion. Conversely, the computer may interrupt the process by identifying that the form is incompatible and independently changes the form to one that it has determined to be "good".

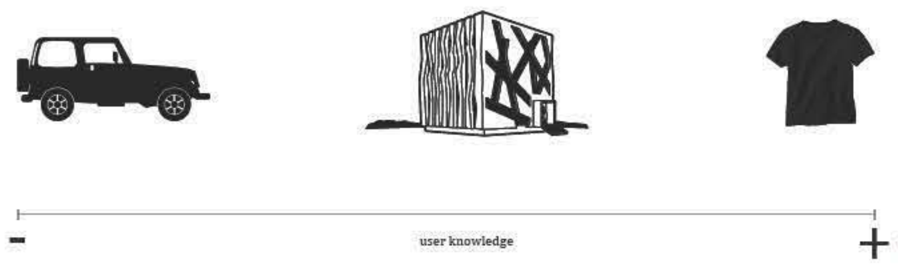
Figure 35, also redrawn from *The Architecture Machine* illustrates the organization of sequential and temporal events in URBAN5. The take-away from this diagram is how the temporal events occur differently between man and machine. The diagram implies, but does not explicitly reference a nonlinear design process that must occur if there is any type of incompatibility.

URBAN5 does not in any way reference the end user of the built form, but I illustrate it here to provide a stark contrast to Negroponte's theories in *Soft Architecture Machines* which was published several years after "The Architecture Machine". In it, Negroponte takes an aggressive stand on user participatory design, supporting the theories made by Friedman and advancing them to the point at which architects are completely removed from the architectural design process.

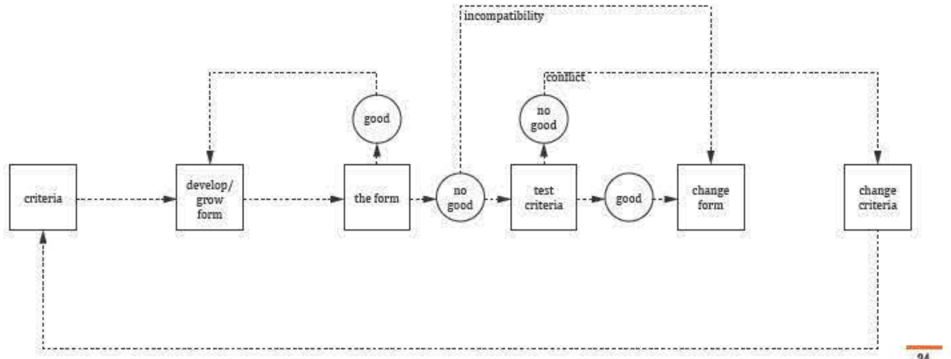
Negroponte states, "The underlying assumption of user participation is that individuals and small groups know what they want or, at least, can learn what they want. The concept further assumes that they can apply this understanding in concert with a 'competence' to realize designs for the built environment."[24]

To illustrate this point Negroponte places architecture between clothing and automobiles in terms of a user's ability to design the product. He points out that many are not familiar enough with combustion and mechanics to design our own cars, but we if we wanted we could easily design our own clothes.

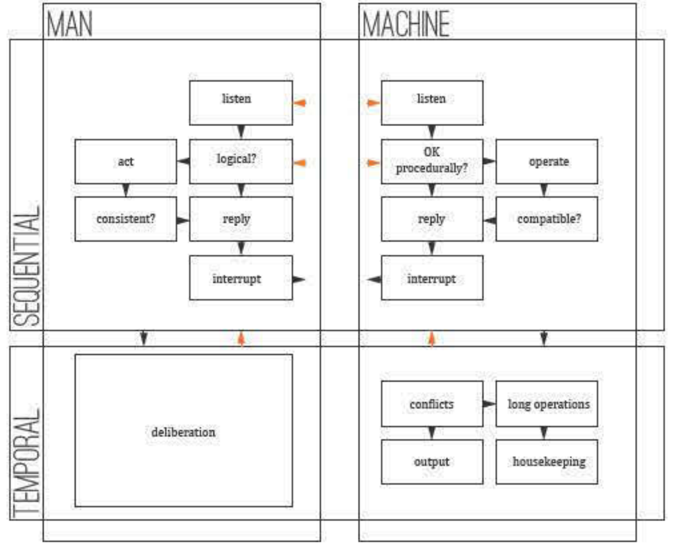
He proposes that a computer interface could provide the necessary amount of competence that would enable the user to create architecture by themselves. Different from Friedman's Flatwriter, Negroponte's solution involves a "learning period" in which the machine would "ask telling and revealing questions and attempt to understand what you mean."[24]



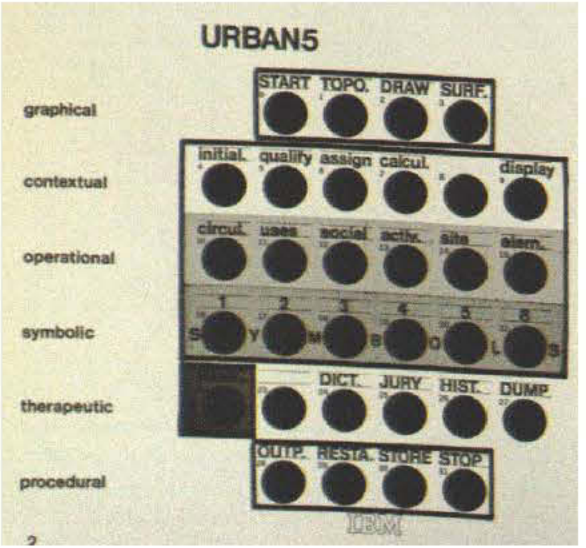
33.

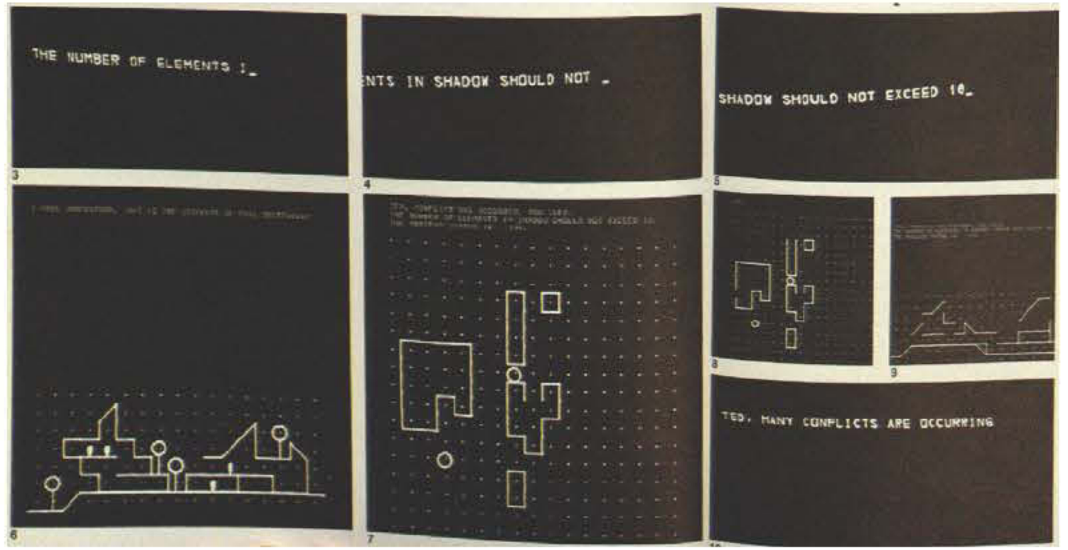


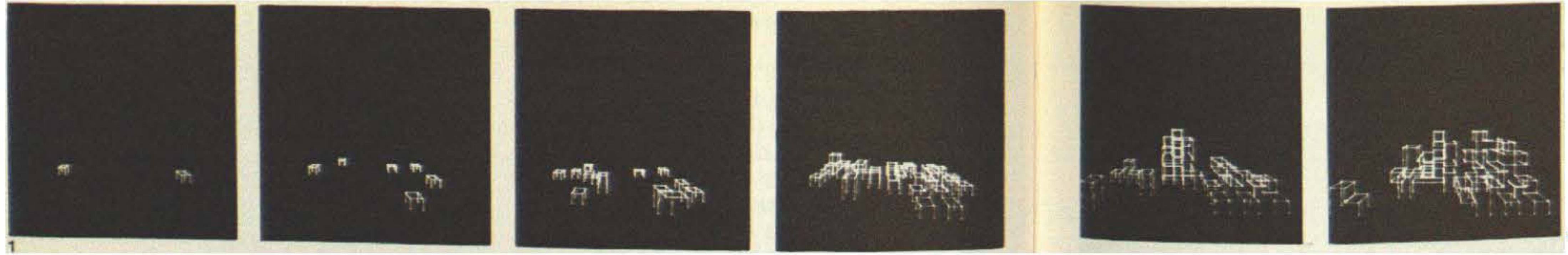
34.

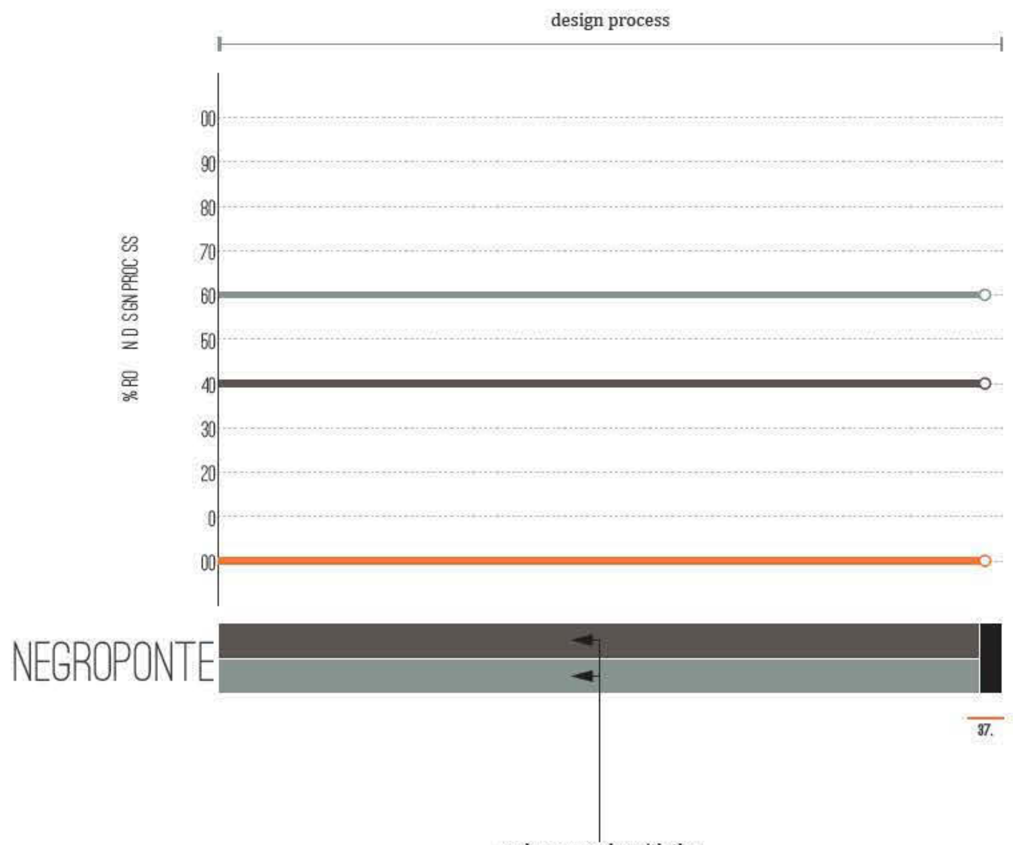


35.









ARCHITECT
USER/CLIENT
COMPUTER/COMPUTING PROCESS
END PRODUCT

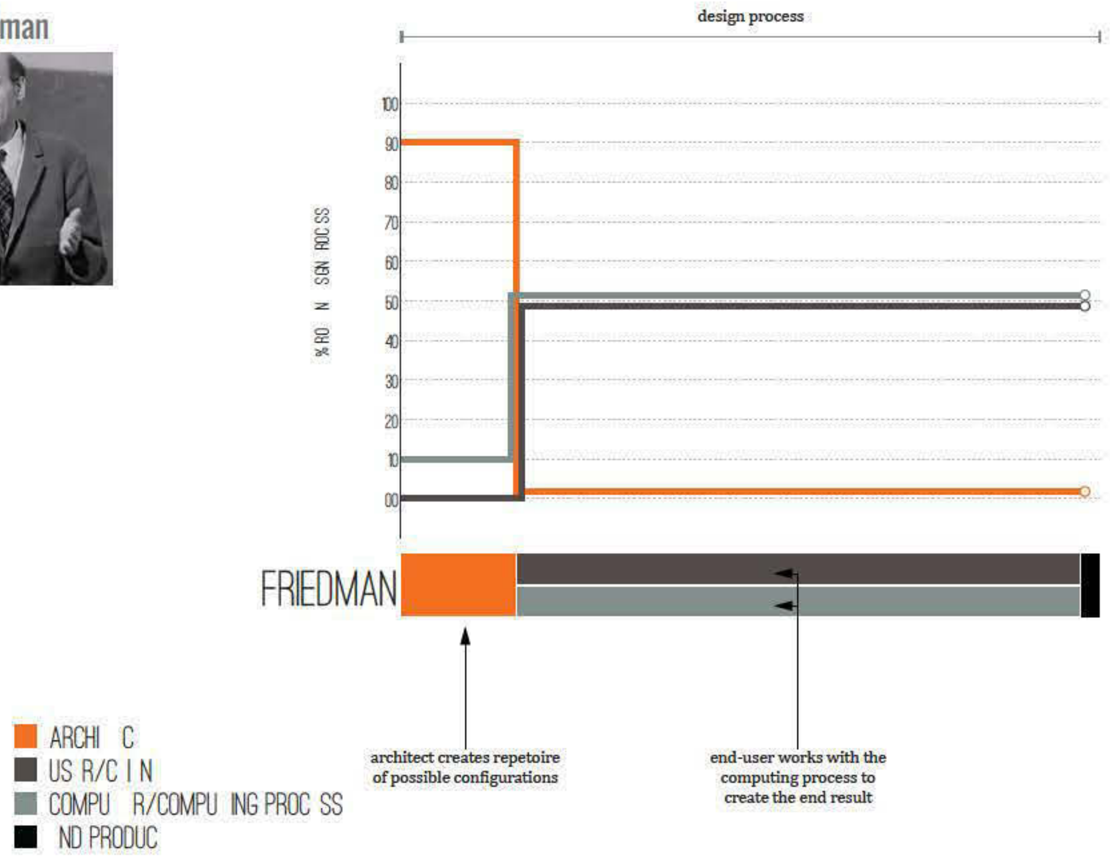
end-user works with the computing process from beginning to end to create the end result

architect is removed from the process

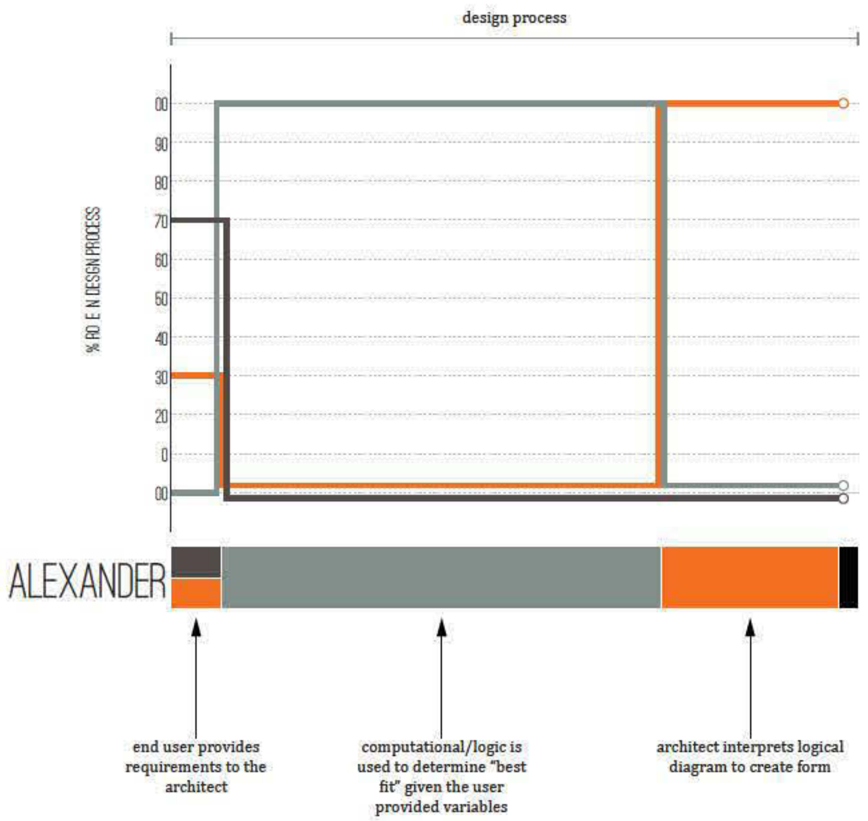
Figure 34. Negroponte, Nicholas. 1972. *The Architecture Machine*. Cambridge, Mass.: MIT Press.

Figure 35. Diagram illustrating the roles of the architect, computer, user, throughout the design process using URBAN5.

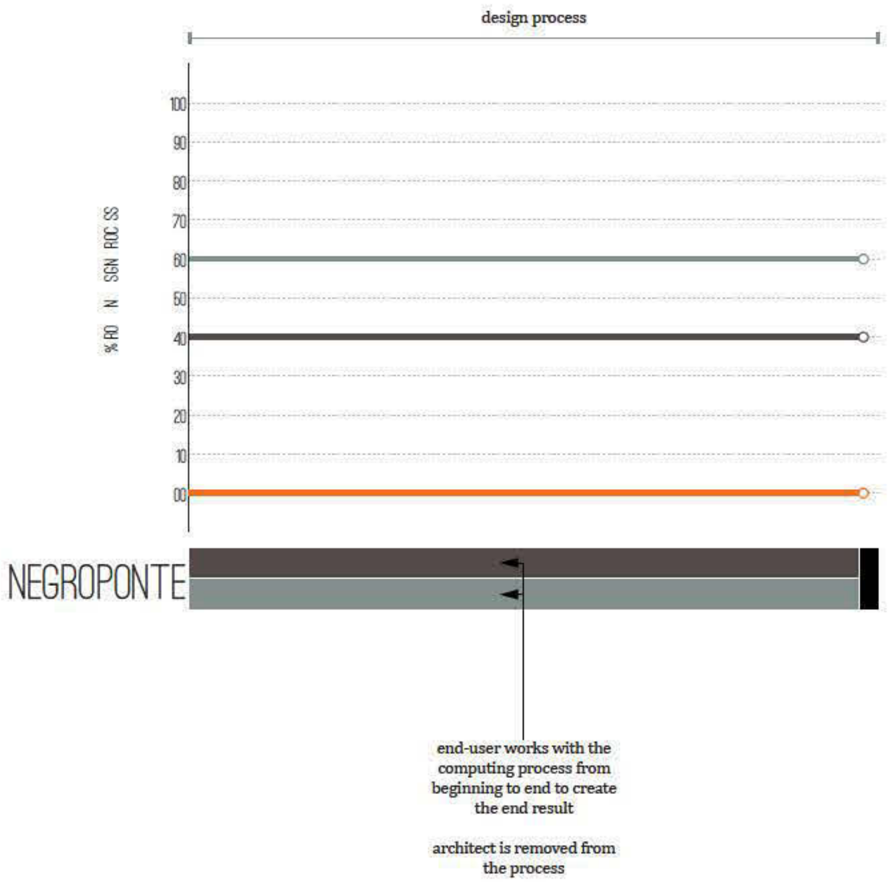
Yona Friedman



Christopher Alexander



Nicholas Negroponte



PROGRAMMING

Looking at these projects the way that I have, I see it very much as a mathematical programming method that fit in constraints, but it was not a computing model that was made for constraint-based work. Instead, it was made for order of operations that these architects had to then shoehorn in constraint-based logic.

To elaborate on two terms I just used, mathematical programming and constraint programming. With mathematical programming, a predefined sequence of steps is defined to reach a particular result (figure 38).

While these steps can have conditional elements, the end solutions are already figured out and the design can only be considered "finished" when the sequence of steps is complete. This requires that the end product is predetermined before beginning the design process.

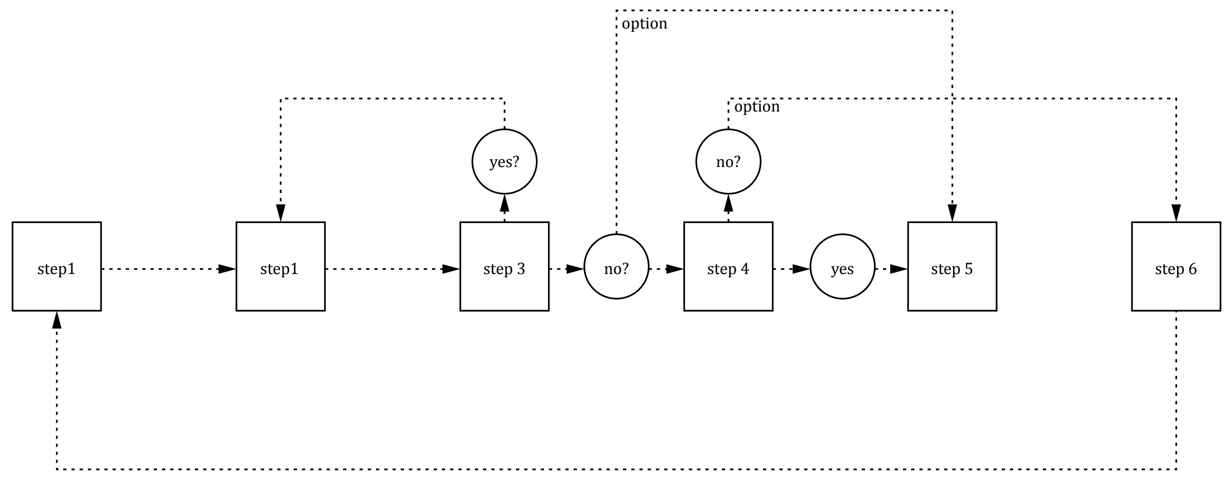
This is problematic because architectural design is not a mathematical. The design process is not a prescribed sequence; it is iterative and non-linear. The solution to a design is not the end result but rather some desirable point in the process when we feel that all our criteria have been met. "We do not find the solution to a set of design specifications; we find one solution out of many alternatives."[25]

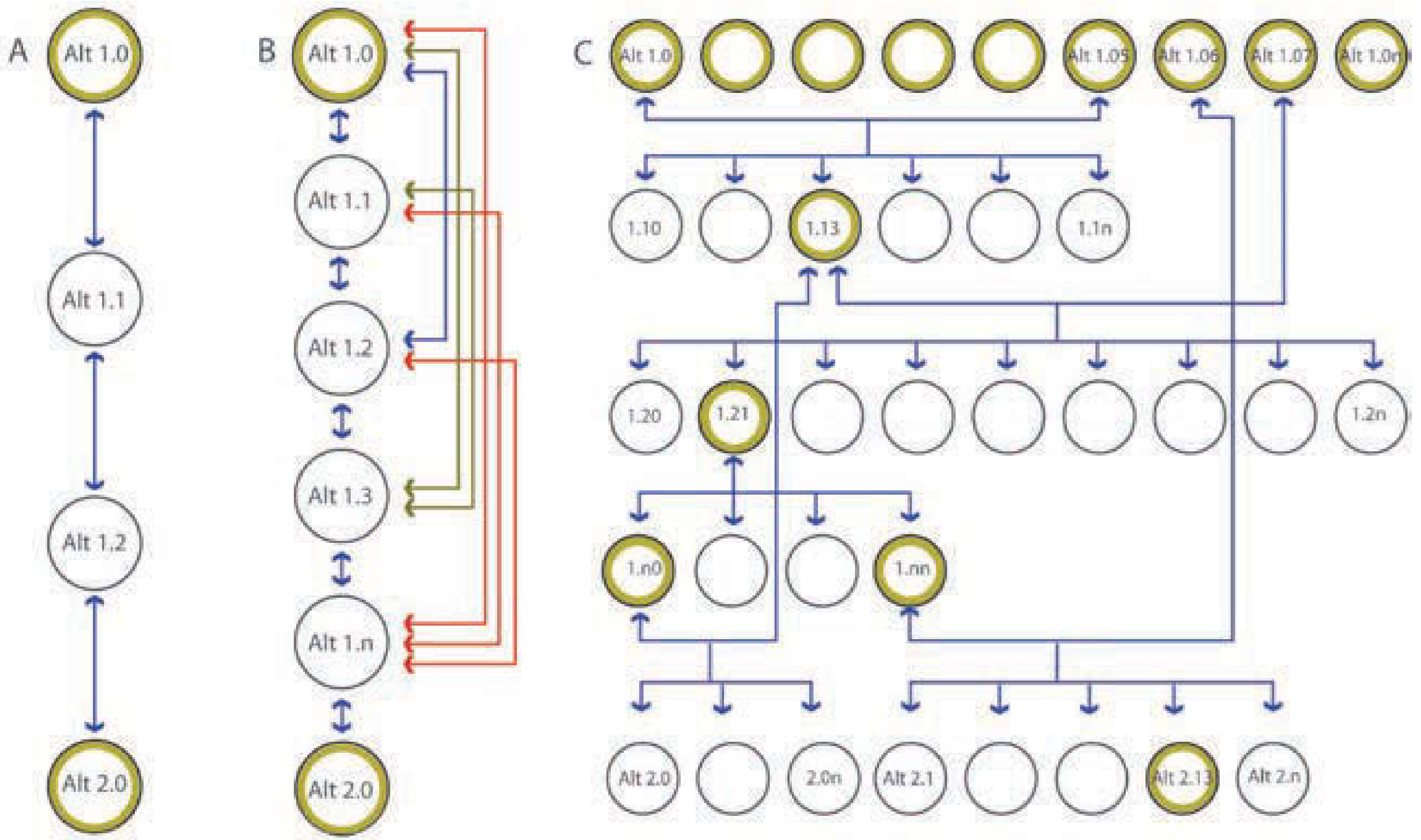
Constraint programming differs from mathematical programming and akin to architectural design in that, it is non-linear and therefore does not specify a sequence of steps. A constraint is a rule. As the name suggests, constraint programming considers any number of constraints, or rules, which must all be satisfied to achieve a solution. It does not propose the process for meeting the constraints. The collection of constraints indicates the boundaries to an infinite solution space.

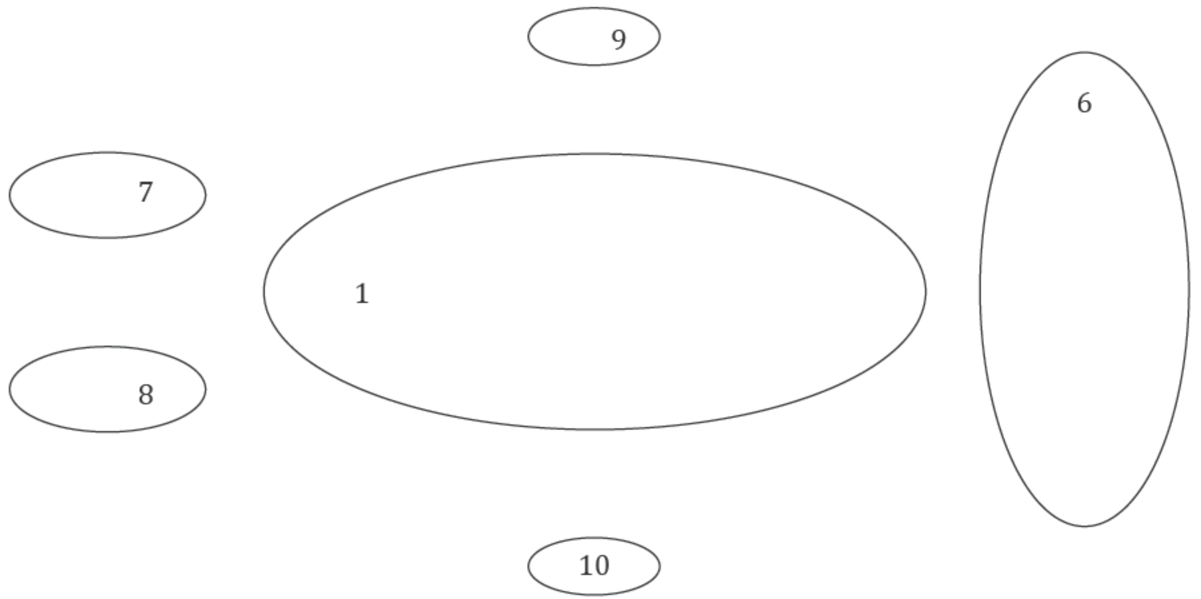
Regardless of the number of constraints and how they are related, there is still a possible solution

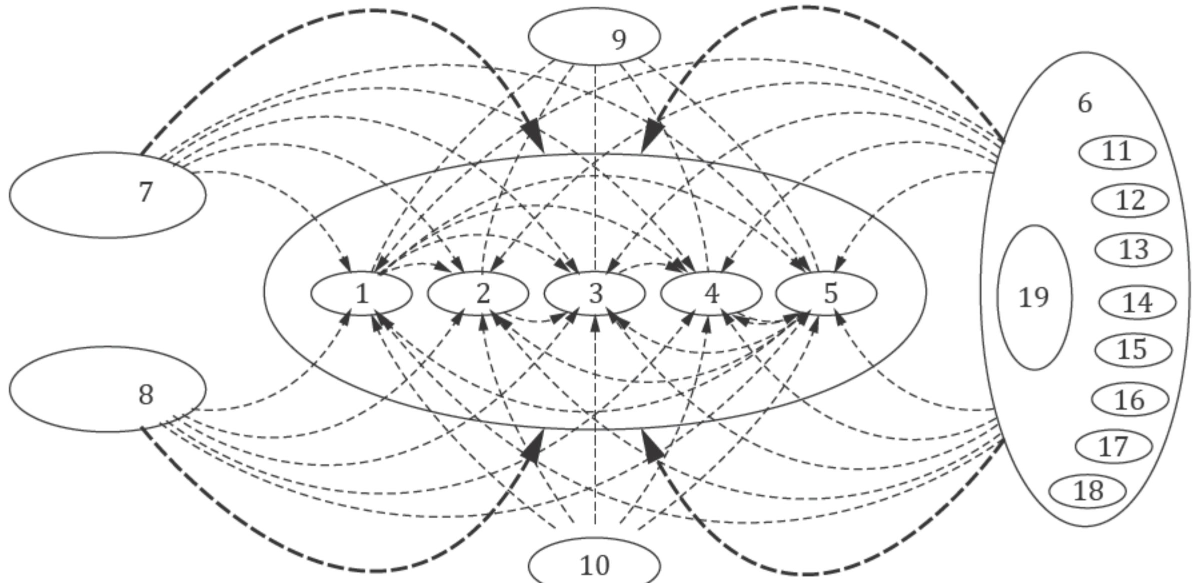
As Alexander suggested, finding the solution is made by compromising between any number of elements within a given solution space.

Mathematical Programming









Constraint = **50% of box must be orange**

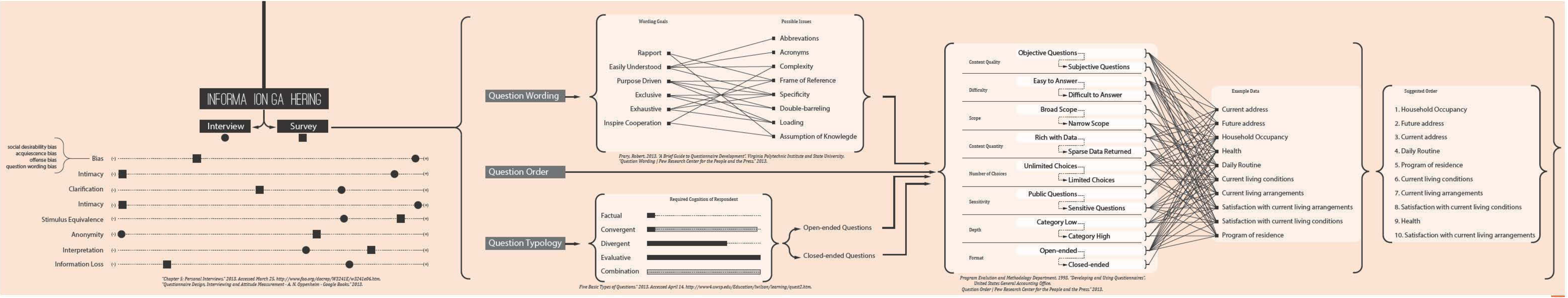
Soutions =

INFORMATION GATHERING

If the constraints are controlling the limits of design, it is imperative that the information we receive is accurate. Traditionally an architecture-led home design usually begins with some form of meeting or interview between the architect and the future occupant of the house. However, studies into information gathering reveal that the interview is the least effective way to obtain information due to bias. "Respondents give answers that they think the interviewer wants to hear, rather than what they really feel and/or... the respondent may be tempted to answer in a way that gives him/her credibility and limits embarrassment in the eyes of onlookers, rather than giving a truthful reply."[26] This is an unavoidable reality of a personal interview process. It is therefore difficult to assume that the constraints derived from this process are as accurate as they could be. Specifically, in the case of architecture the problem is enhanced by the fact that population at large is unaware of what architects do, what is considered when designing a building or why we consider all of that information

To formulate a more accurate way of collecting information from clients I researched the science of survey methodology. Within this, there are three main prerogatives: question wording, question order, and question typology.

In summary there are ways of asking certain types of questions that are worded in a particular way and presented in a particular order that have the greatest potential of yielding accurate and truthful responses from the future occupant of a home.



INFORMA ION GA HERING
Interview
Survey
social desirability bias
acquiescence bias
offense bias
question wording bias
Bias
Intimacy
Clarification
Intimacy
Stimulus Equivalence
Anonymity
Interpretation
Information Loss
"Chapter 5: Personal Interviews." 2013. Accessed March 25. http://www.fao.org/docrep/W3241E/w3241e06.htm.
"Questionnaire Design, Interviewing and Attitude Measurement - A. N. Oppenheim - Google Books." 2013.
Question Wording
Question Order
Question Typology
Wording Goals
Possible Issues
Rapport
Easily Understood
Purpose Driven
Exclusive
Exhaustive
Inspire Cooperation
Abbrevations
Acronyms
Complexity
Frame of Reference
Specificity
Double-barreling
Loading
Assumption of Knowlegde
Frary, Robert. 2013. "A Brief Guide to Questionnaire Development". Virginia Polytechnic Institute and State University.
"Question Wording | Pew Research Center for the People and the Press." 2013.
Required Cognition of Respondent
Factual
Convergent
Divergent
Evaluative
Combination
Open-ended Questions
Closed-ended Questions
Five Basic Types of Questions." 2013. Accessed April 14. http://www4.uwsp.edu/Education/lwilson/learning/quest2.htm.
Content Quality
Objective Questions
Subjective Questions
Difficulty
Easy to Answer
Difficult to Answer
Scope
Broad Scope
Narrow Scope
Content Quantity
Rich with Data
Sparse Data Returned
Number of Choices
Unlimited Choices
Limited Choices
Sensitivity
Public Questions
Sensitive Questions
Depth
Category Low
Category High
Format
Open-ended
Closed-ended
Program Evalution and Methodology Department. 1993. "Developing and Using Questionnaires".
United States General Accounting Office.
Question Order | Pew Research Center for the People and the Press." 2013.
Example Data
Current address
Future address
Household Occupancy
Health
Daily Routine
Current living conditions
Current living arrangements
Satisfaction with current living arrangements
Satisfaction with current living conditions
Program of residence
Suggested Order
1. Household Occupancy
2. Future address
3. Current address
4. Daily Routine
5. Program of residence
6. Current living conditions
7. Current living arrangements
8. Satisfaction with current living conditions
9. Health
10. Satisfaction with current living arrangements

INTERFACE & OPTIMIZATION PRECEDENCE

The BVO model employs constraint programming (CP) instead of mathematical programming for the search of feasible and optimal solutions. This is a crucial element of the project. In mathematical programming, the program code consists of a sequence of steps that are followed in order to achieve a result. CP, on the other hand, uses a declarative programming environment. "A Constraint Program is not a statement of a problem as in mathematical programming, but is rather a computer program that indicates a method for solving a particular problem."[27]

Recalling Alexander's theory on defining a design problem through a identifying the negative requirements, it is easy to the relation to CP.

The differences in programming languages relate directly to the architectural design processes I am illustrating. Mathematical programming is, by its very nature, a linear process; the variables are defined, the sequence of steps is identified, and one best answer is found. CP, on the other hand, is non-linear because is continually searching for an optimized solution based on a set of constraints. There is not a unique, defined solution, but rather an optimized discovery that is the result of multiple iterations within the constraints of the design domain.

It is obviously difficult to say which routes architects follow while designing. It is therefore complicated to code a set of steps that describe how a design problem could be solved by a machine. Referring to what was said before, a different programming paradigm that specifies a set of constraints that must be met without stating how to achieve this task.

In 2010, Yasha Grobman published an article in the International Journal of Architectural Computing entitled "Non-Linear Architectural Design Process". In it, he touches on several of the elements I previously highlighted but delves deeper into the differences between linear parametric design and a non-linear process. Grobman states,

"The main difference between the linear parametric design process and the non-linear process, besides the obvious ability to generate and work with several design alternatives, has to do with the ability of the nonlinear algorithm to generate new alternatives deriving from both single and multiple initial alternatives. This allows the designer to combine successful alternatives from different sub-stages in the generation process.

The idea of multiple design solutions has been discussed widely in traditional design thinking and cognition discourse. Some examples are the discussion on parallel lines of thought by Lawson, the discussion on top-down and bottom-up approaches within the space problem by Rowe and Alexander's procedural design method described in his seminal book 'Notes on the Synthesis of Form'. However, these approaches and methods still fall within the realm of linear design. Although, the possibility to go back and forth during the design process is mentioned and discussed in these texts, they do not discuss nor mention the option of combining ideas from various stages of the design process as suggested in the nonlinear design process. This can be explained by the connection of nonlinear design to computers, which were not widely used for design when these ideas were developed."[28]

In developing his program, Schoch relied on the findings of T.M. Locher who stated that, "the use of a mathematical description to characterize a design problem implies the following hypotheses:

(1) Architectural design is affected by rules. (2) Rules can be used to constrain the solution space of a design problem. (3) Provided that constraints and objectives are specified by the architect, computers can extend the number of feasible solutions for a design problem.[29]

Furthermore, it is possible to classify the following assumptions as integral components of an optimization model. (1) The design solution has to meet specific requirements [constraints]; (2) the design has to strive for specific goals [objectives];(3) there are choices that might meet the constraints and objectives [design variables].

The optimization model itself consists of a given number of variable and constant parameters, one or more objectives, as well as a fluctuating number of constraints. Each object that belongs to the model can be accessed and altered by the use of parameters. A room, for example, is an object with geometric parameters such as length, width, and height. Objects can also imply alphanumerical parameters such as their occupancy or neighborhood. Parameters are defined in the form of variables or constants, whereas variables can be used as inputs for the optimization process. Responses result from the composition of other variables. If a variable is changed during the optimization process, dependent variables will be changed as well. Inputs and Responses are often named Optimization Variables. [30]

These variables form the basis of constraints and objective functions. Both must be functions of one or more optimization variables. Within an architectural problem domain, a response variable could be the area occupied by a specific room. Through multiplication of two input parameters (width and length), a response variable would be rendered. It is of primary interest that suchlike parameters generate serious problems for the optimization process due to their nonlinear form. Once the design problem is stated in form of design variables, constraints, and objectives, the parameters will be passed to the optimization engine, which tries to find a feasible solution to the problem. A programming language that supports this paradigm and that was used herein is OPL (Optimization Programming Language), which was developed in 1995.[30]

The principle of the geometric model adopted is the representation of rooms as rectangular units. Michalek [4] demonstrated this concept in his work on architectural layout planning. In contrast to his concept, a geometric representation was chosen that describes a rectangular unit through a reference point, a length, and a width dimension. [30]

Constraints were taken from this work that describe the location of a unit inside another (Force Inside), the intersection of two units (Prohibit Intersection), the location of a unit on the border of another unit (Force To Border), the connection of two units (Force Connection), the location of a unit on the outside of another unit (Force Outside), as well as the prohibition of a connection between two units (Prohibit Connection). [30]

Various design constraints (e.g. aspect ratio, symmetry) that refer to subjective rules were implemented. These design constraints as well as constraint combinations make it possible to extend the architect's ability to intervene in the creative process of automatic layout planning.

The use of constraint combinations, for example, led to a new constraint that made it possible to extend the geometric model to non-rectangular units. These so-called Void Units accommodate complex shapes that must not be specified differently from other units, according to their geometrical measures. [30]

In figure 49, a rectangular floor plan with an area of 900 square meters and side lengths of 30 by 30 meters is shown. Satisfying a large number of additional conditions, an arrangement of the nineteen areas of the room program had to be found with the sum of the areas of slots 1 through 8 equaling the total area of the building floor plan and with slots 11 through 19 arranged within slot 6 (figure 50). [31]

In attempting to determine what occurs in the computational precursor stage, I looked to research that is more contemporary. In 2011, Martin Schoch of Shinawatra University in Bangkok, Thailand published an article entitled, Building-volume designs with optimal life-cycle costs. In it, he describes a methodology in which a computational decision-support system would address problems associated with missing quantitative information.

Using a constraint programming language, the BVO model enables designers to find design solutions that offer cost-effectiveness. Minimizing LCC, it determines optimal-volume dimensions, number of floor levels, building orientation and opening ratios of exterior surface-areas while satisfying site criteria, building-code regulations and design constraints such as suggested floor-area usage boundaries or building depth. Further, through its three-dimensional building-volume visualization of optimal or feasible solutions, the BVO model allows for a comprehensive understanding between its implemented optimization strategies and its resulting effect on the continuously improving building-volume shape. [32]

Schoch determined that the lifecycle costs of a building are the result of the summation of four costs: Energy Costs (EC), Construction Costs (CC), Operation and Maintenance Costs (OMC), and Repair and Renovation Costs (RRC). Thus by adding the present value of these costs one can determine the present value of the total Life Cycle costs of a building. By tying these costs to the aforementioned variables, Schoch was able to develop a software application that evaluates the quantitative data and proposes an overall building volume that is optimized with regards to lowering the life cycle cost of the building. [33]

"For implementation, the BVO model facilitates two existing software applications. The optimization of building-volumes using CP techniques is realized with ILOG OPL Studio 6.1.1 [25]. For the visualization of its results, Processing, an open-source programming environment for data presentation and visualization is used [26].When conducting experiments, the solution-data is recorded; successful optimization runs are then visualized, allowing for visual examination of all feasible and optimal solutions, as illustrated in Figure 4." [33]

"The results of the BVO model testing confirmed the assumption that the CP engine solver continuously improves the found solutions. An optimal solution could be found within a practical period of less than three minutes with the range of the allowable building-volume opening ratio limited to 40 - 60%.The generated volume solutions of the test runs satisfied the model constraints and remained within the theoretical building-volume. Repeated optimization runs with similar setups concluded with the same optimal objective value. Yet, earlier attempts showed that search time could significantly increase when the specifications of decision variables are inconsiderably high. For example, by allowing the opening ratio to use a range between 0 - 100 %, the search space increases unnecessarily. The model results thus indicate that thoughtful calibration of its decision variables is required." [33]

The BVO model is a promising tool in the development of cost effective buildings, but it is geared for use exclusively by architects and construction professionals and does nothing to facilitate a design process that incorporates the future user. In addition, it fails to consider how this breakthrough analysis tool could be used to assist designers throughout the design process.

Friedman and Negroponte propose methodologies that reduce and even eliminate the role of the architect in the architectural design process, claiming that it is irresponsible to let an architect dictate a design because the architect does not have the suffer the consequences of poor design choices. The future is the best person to create space, heavily proposing "architecture by yourself".
Alexander proposes a methodology in which the architect utilizes logic and set theory to determine the best fit for design decisions, employing a primitive version of constraint programming. Arguing in an opposite fashion that not even an architect, and certainly nor the future user is capable of making competent design decisions without the use of a logic based computation process to evaluate criteria. Each of the aforementioned theories suffer from being a linear process, however, contemporary technology and programming methods such as constraint and optimization programming create an opportunity to revisit and reapply these theories within the framework of a modern system.

ONE

Architectural design is affected by rules

TWO

Rules can be used to constrain the solution space

THREE

Computer can extend the number of feasible options

ONE

Design solution has to meet specific requirements

(Constraints)

TWO

The design has to strive for specific goals

(Objectives)

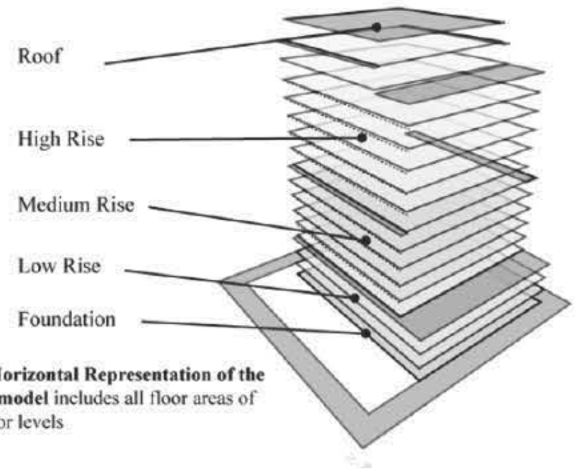
THREE

There are choices available that meet the constraintsand objectives

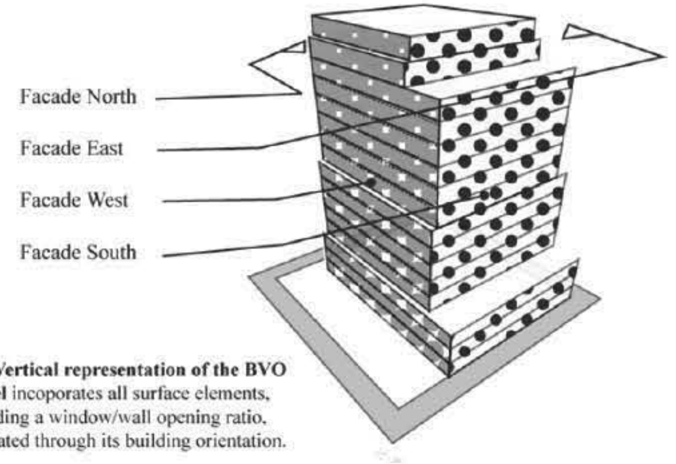
(Design Variables)

BUILDING-VOLUME OPTIMIZATION

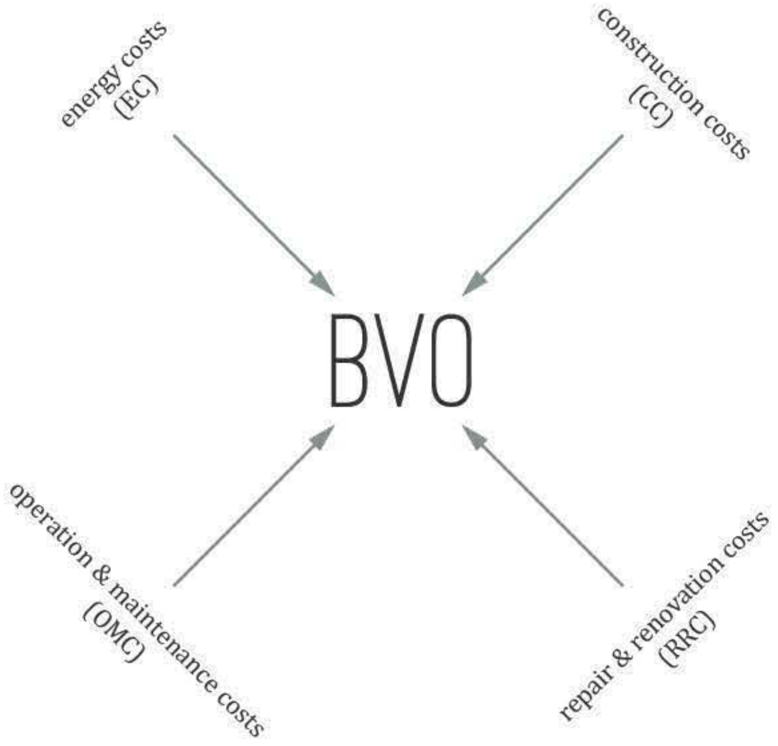
a computational decision-support for designers addressing problems
associated with missing quantitative design aids
during the early architectural design phase



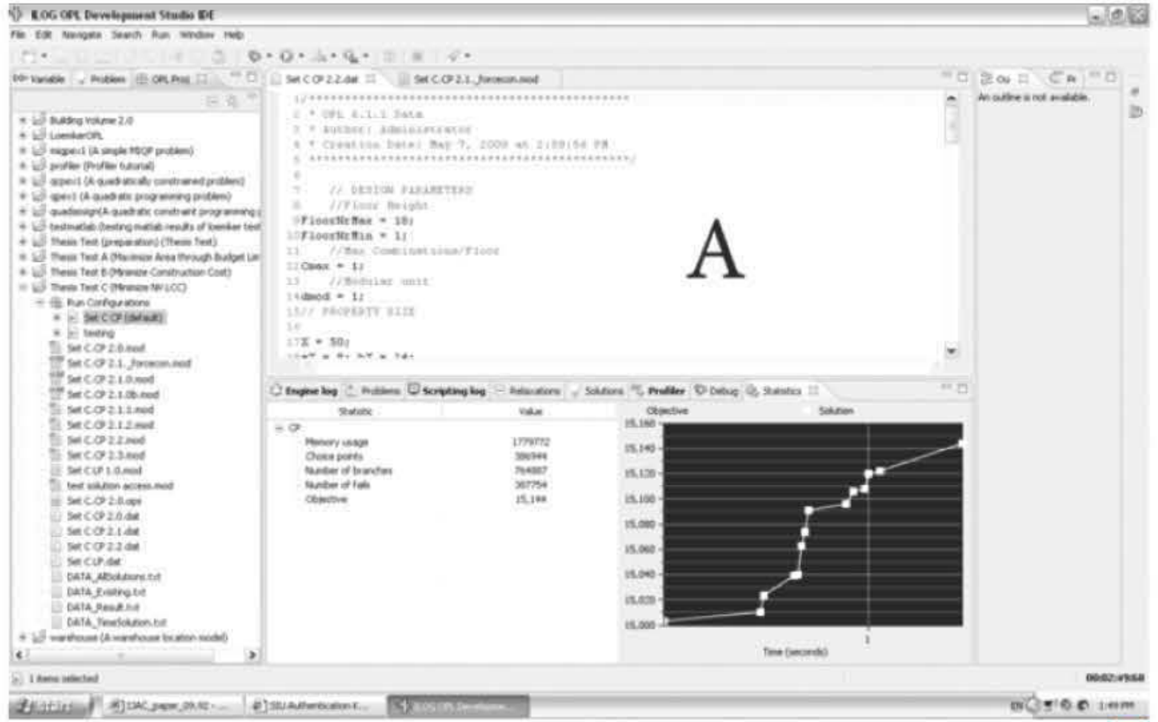
The Horizontal Representation of the BVO model includes all floor areas of all floor levels



The Vertical representation of the BVO model incoporates all surface elements, including a window/wall opening ratio, seperated through its building orientation.

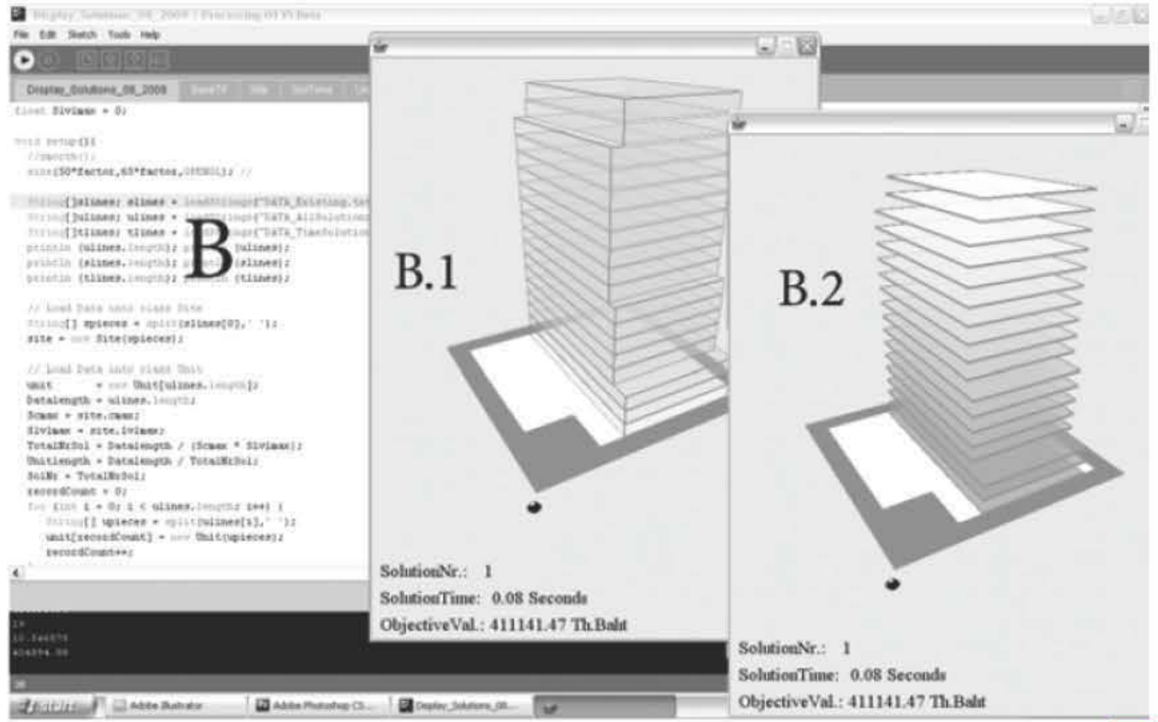


$$\textit{Minimize: } LCC_{PV} = CC_{PV} + EC_{PV} + OMC_{PV} + RRC_{PV}$$



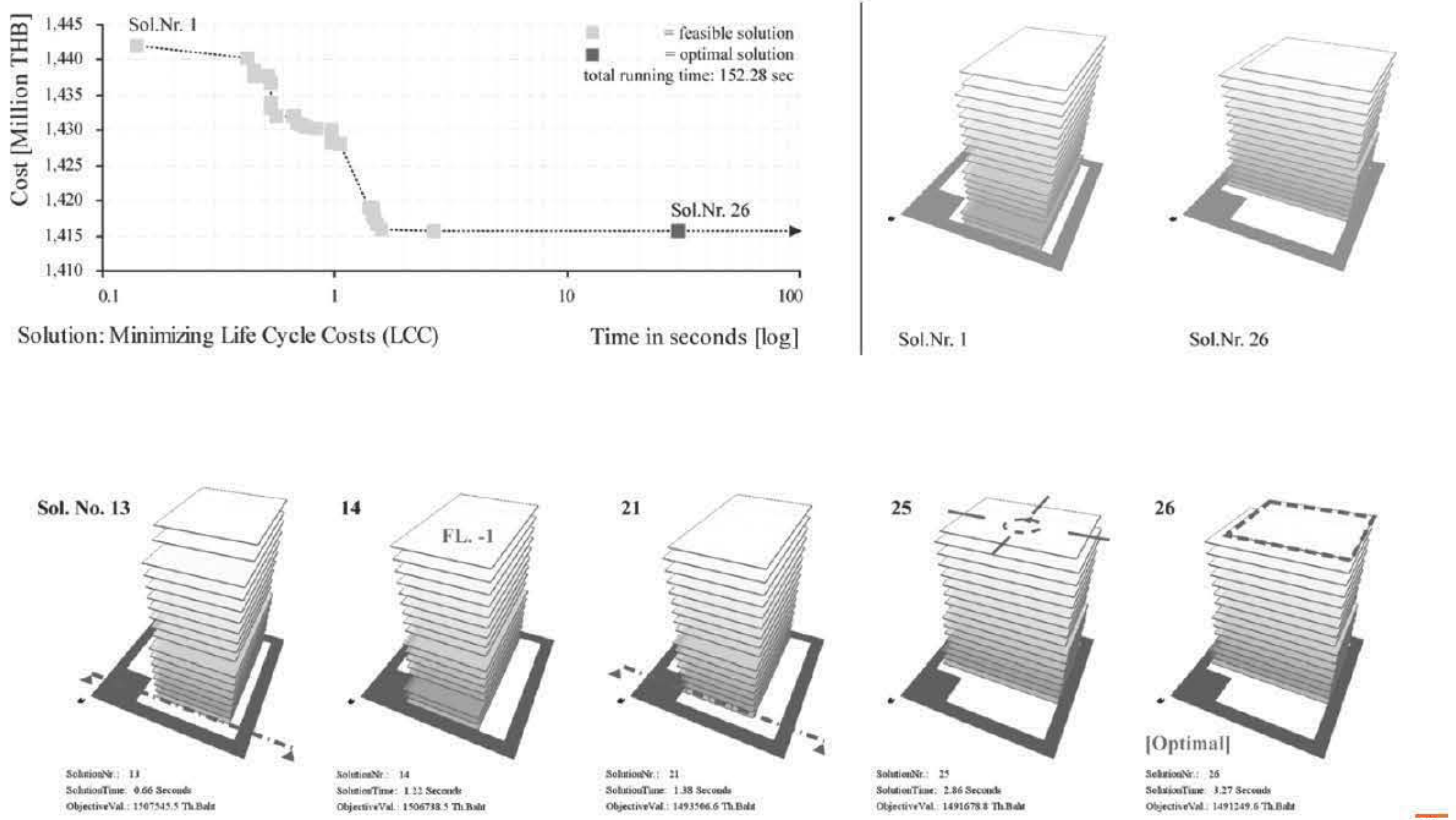
ILOG OPL STUDIO 6.1.1

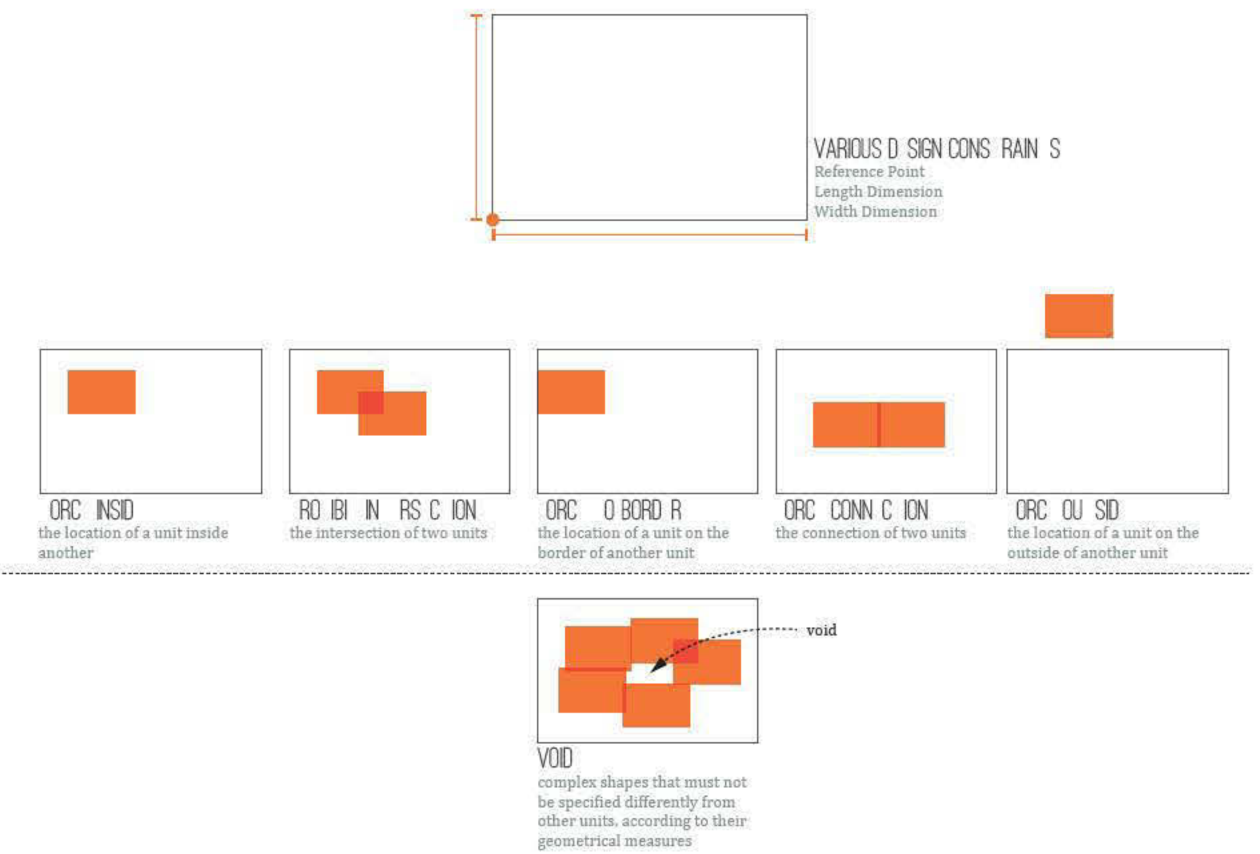
The optimization of building-volumes using CP techniques

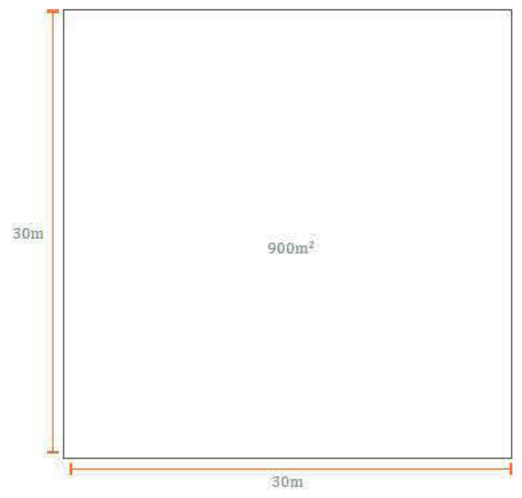


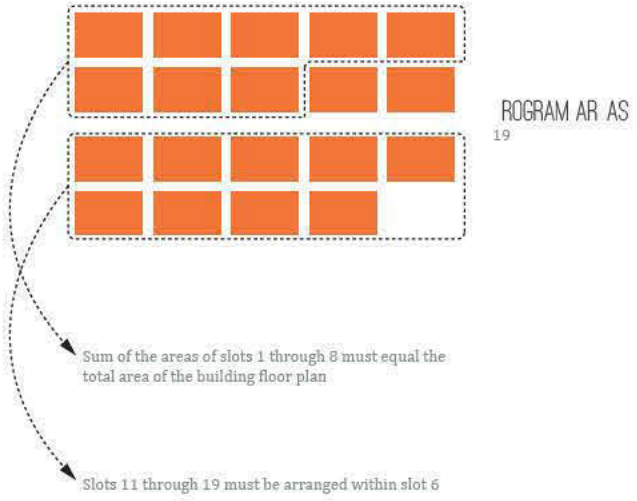
PROCESSING

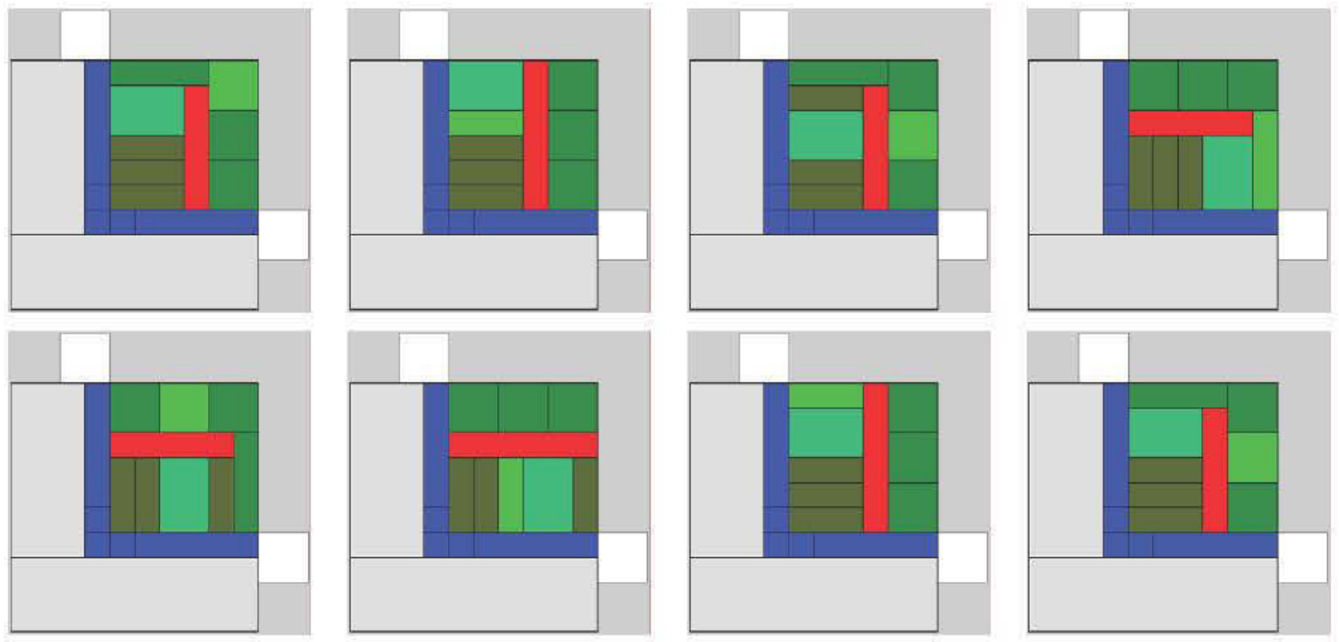
Visualization of ILOG OPL Studio results




VARIOUS D SIGN CONS RAIN S
Reference Point
Length Dimension
Width Dimension
ORC INSID
the location of a unit inside another
RO IBI IN RS C ION
the intersection of two units
ORC O BORD R
the location of a unit on the border of another unit
ORC CONN C ION
the connection of two units
ORC OU SID
the location of a unit on the outside of another unit
void
VOID
complex shapes that must not be specified differently from other units, according to their geometrical measures


30m
900m²
30m


ROGRAM AR AS
19
Sum of the areas of slots 1 through 8 must equal the total area of the building floor plan
Slots 11 through 19 must be arranged within slot 6



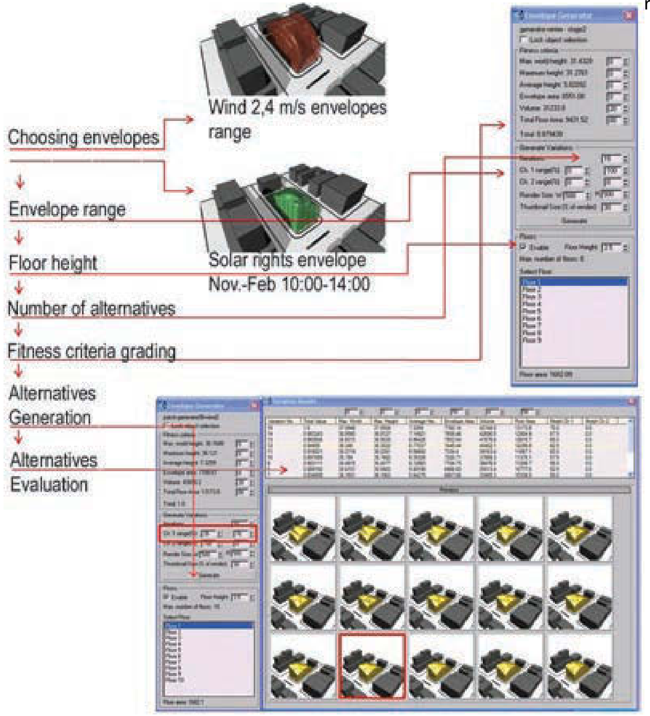
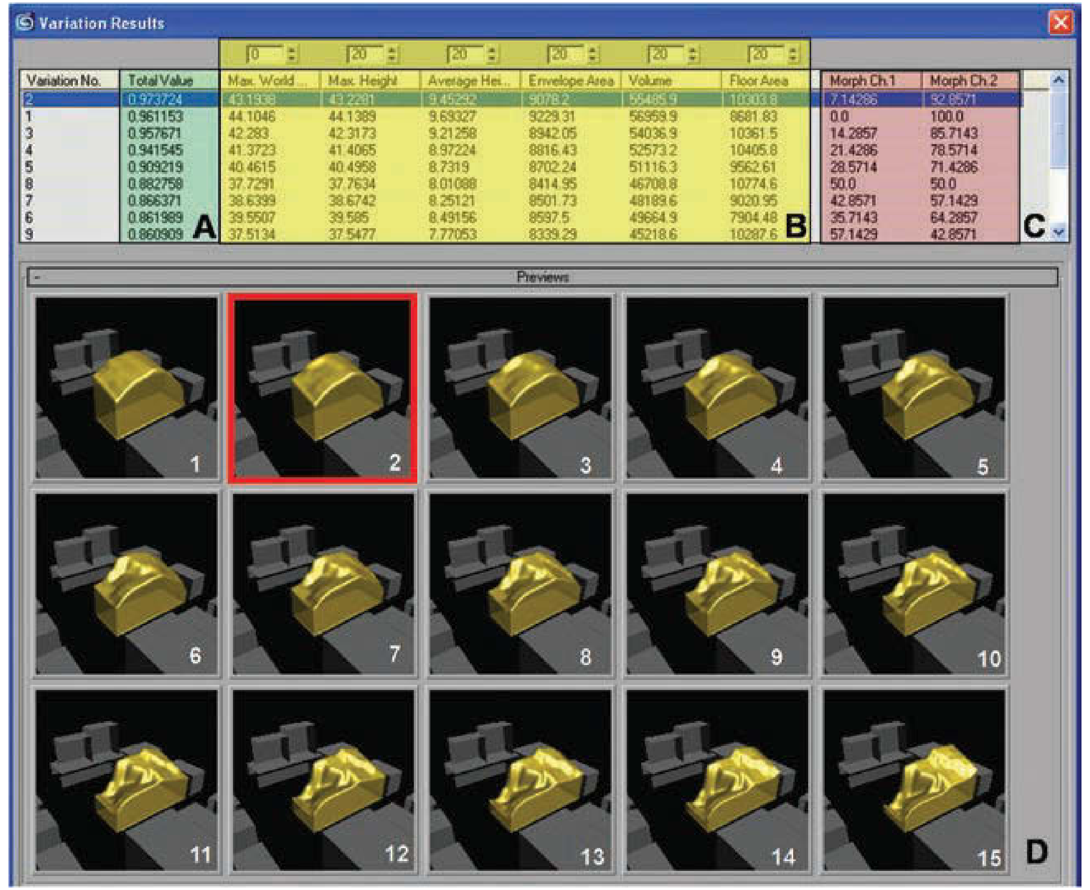


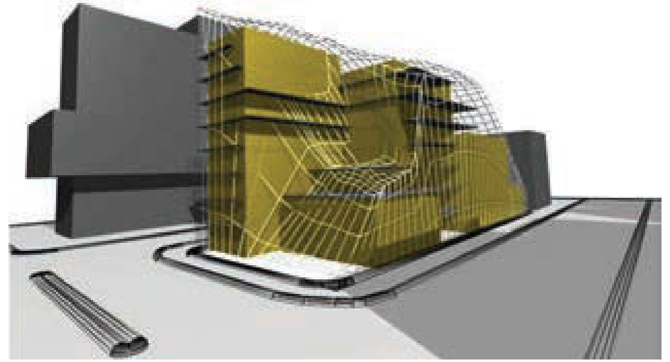
Figure 3: User interface, initial set up and generated alternatives visual interactive catalogue (the alternative with the highest grade in the current fitness settings is marked by a dark/red rectangle).



Variation No.	Total Value	Max. World ...	Max. Height	Average Hei...	Envelope Area	Volume	Floor Area	Morph Ch.1	Morph Ch.2
		0	20	20	20	20	20		
2	0.973724	43.1938	43.2281	9.45292	9078.2	55485.9	10303.8	7.14286	92.8571
1	0.961153	44.1046	44.1389	9.69327	9229.31	56959.9	8681.83	0.0	100.0
3	0.957671	42.283	42.3173	9.21258	8942.05	54036.9	10361.5	14.2857	85.7143
4	0.941545	41.3723	41.4065	8.97224	8816.43	52573.2	10405.8	21.4286	78.5714
5	0.909219	40.4615	40.4958	8.7319	8702.24	51116.3	9562.61	28.5714	71.4286
8	0.882758	37.7291	37.7634	8.01088	8414.95	46708.8	10774.6	50.0	50.0
7	0.866371	38.6399	38.6742	8.25121	8501.73	48189.6	9020.95	42.8571	57.1429
6	0.861989	39.5507	39.585	8.49156	8597.5	49664.9	7904.48	35.7143	64.2857
9	0.860909	37.5134	37.5477	7.77053	8339.29	45218.6	10287.6	57.1429	42.8571

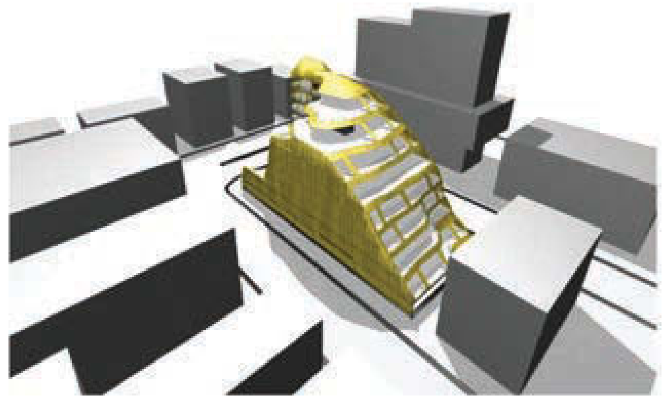
GENPOD VARIATION RESULTS

Visual catalogue of generated alternatives: A-total grade, B-extrinsic criteria's values, C-deviation/adherence to performance envelope (intrinsic criteria), D-generated alternatives (the alternative with the highest grade in the current fitness settings is marked by a dark/red rectangle).



GENPOD - POSSIBLE USE OF THE SELECTED ENVELOPE

Design space



GENPOD - POSSIBLE USE OF THE SELECTED ENVELOPE

Building's initial form

INTERFACE DESIGN & FUNCTIONALITY

It would be impossible to analyze every element of design and critical decision in the design process, therefore the scope of my argument will specifically consider a finite set of three variables: (1) site, (2) enclosure, and (3) materials. These three variables are the topics outlined by David Leatherbarrow in *The Roots of Architectural Invention*. In this text, Leatherbarrow states that every architect will have to consider site, enclosure, and materials at some point in the architectural design process. Moreover, he argues that these elements must be worked out fully before considerations of style and aesthetic are applied. [34]

The interface I have designed is a constraint driven program that (1) indirectly obtains missing design information by asking the user about the quality of the spaces in their home in layman's terms; (2) it serves as an impersonal communication interface between the future occupant of the home and an architect; and (3) it is 3D modeling and design software that uses the information that it gathered to facilitate the design process.

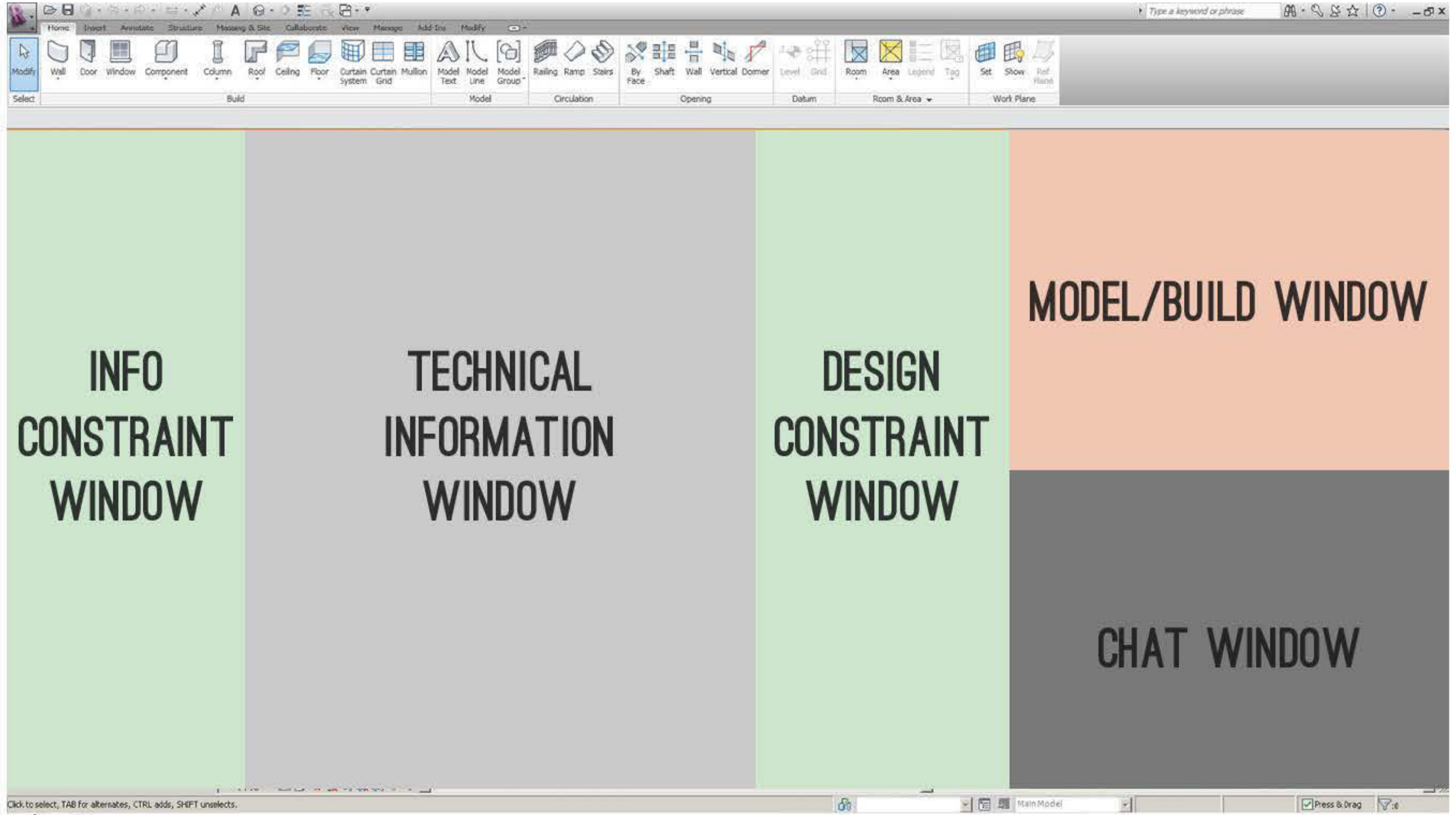
The most troubling aspect of current DIY home design software is the user is typically unaware of the full scope of their design decisions. Maybe it is something simple like a building code violation or maybe it is something more complex like a design feature that hinders passive cooling when that was originally something that the user really wanted Integrating design constraints prevents the user from making uninformed decisions.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are answered by the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. If a client is asked to determine the best orientation for their house, there are dozens of variable that go into making that decision that the average consumer will likely not have the expertise to consider. However, if asked, "do you like sunlight to come into your bedroom in the morning?" that question helps to determine the orientation of the house. It also helps to determine the location of the bedroom in the overall plan, the number of widows or amount of transparency that is present etc.

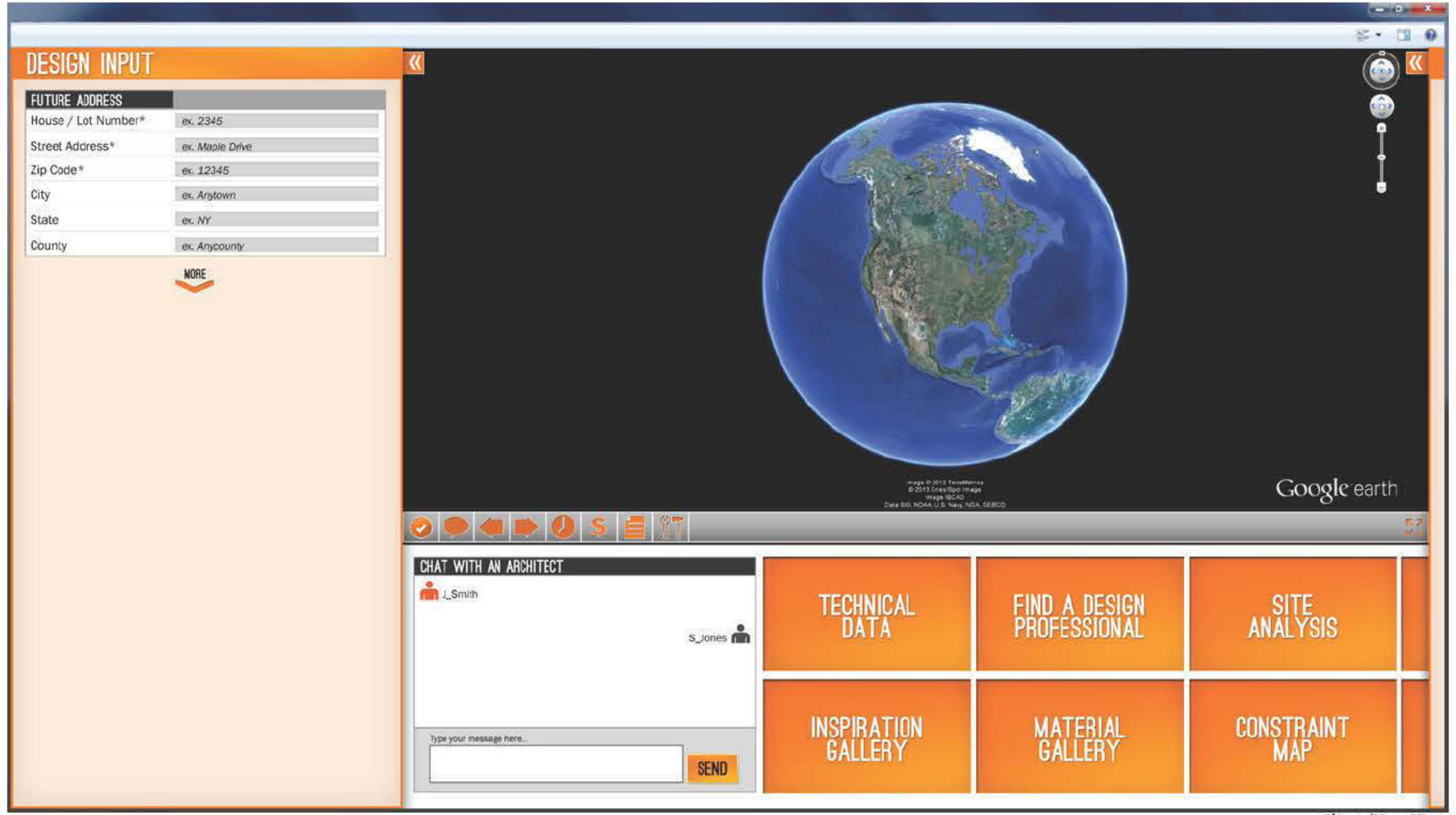
The occupant is the expert in how they want to live; the architect should facilitate the most appropriate design to meet that expectation.

The interface begins by requesting simple, objective information like the property address. From this address, the program can pull in thousands of constraints defined by the building code, zoning codes, and homeowner's association design regulations and it can begin assembling the data for constraints that have not yet been defined by the user such as climate data, topography, soil conditions, etc. From the constraints, the program can formulate that without some type of variance, the volume here is the maximum buildable volume of the house.

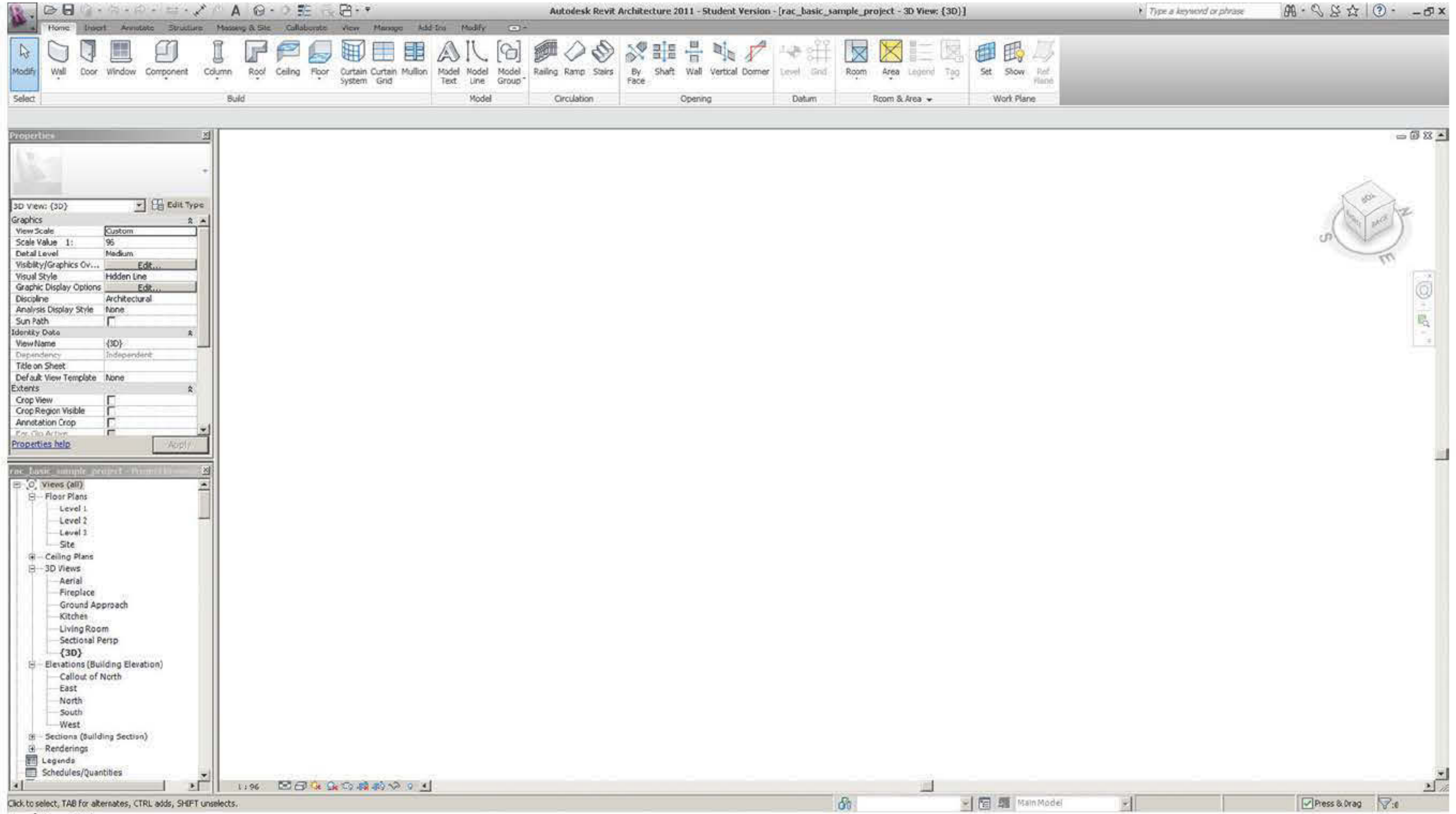
Client/User View



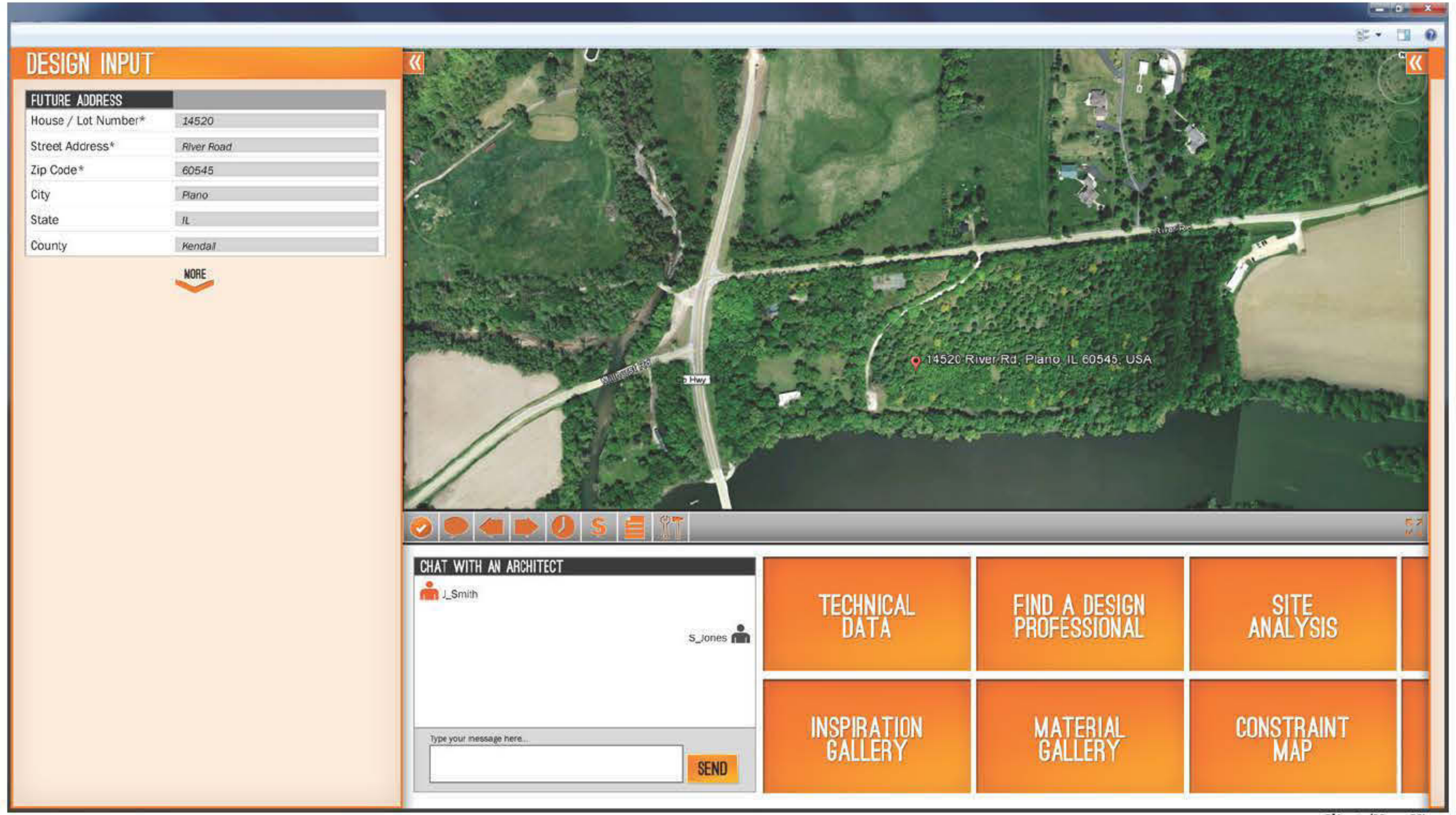
Architect View



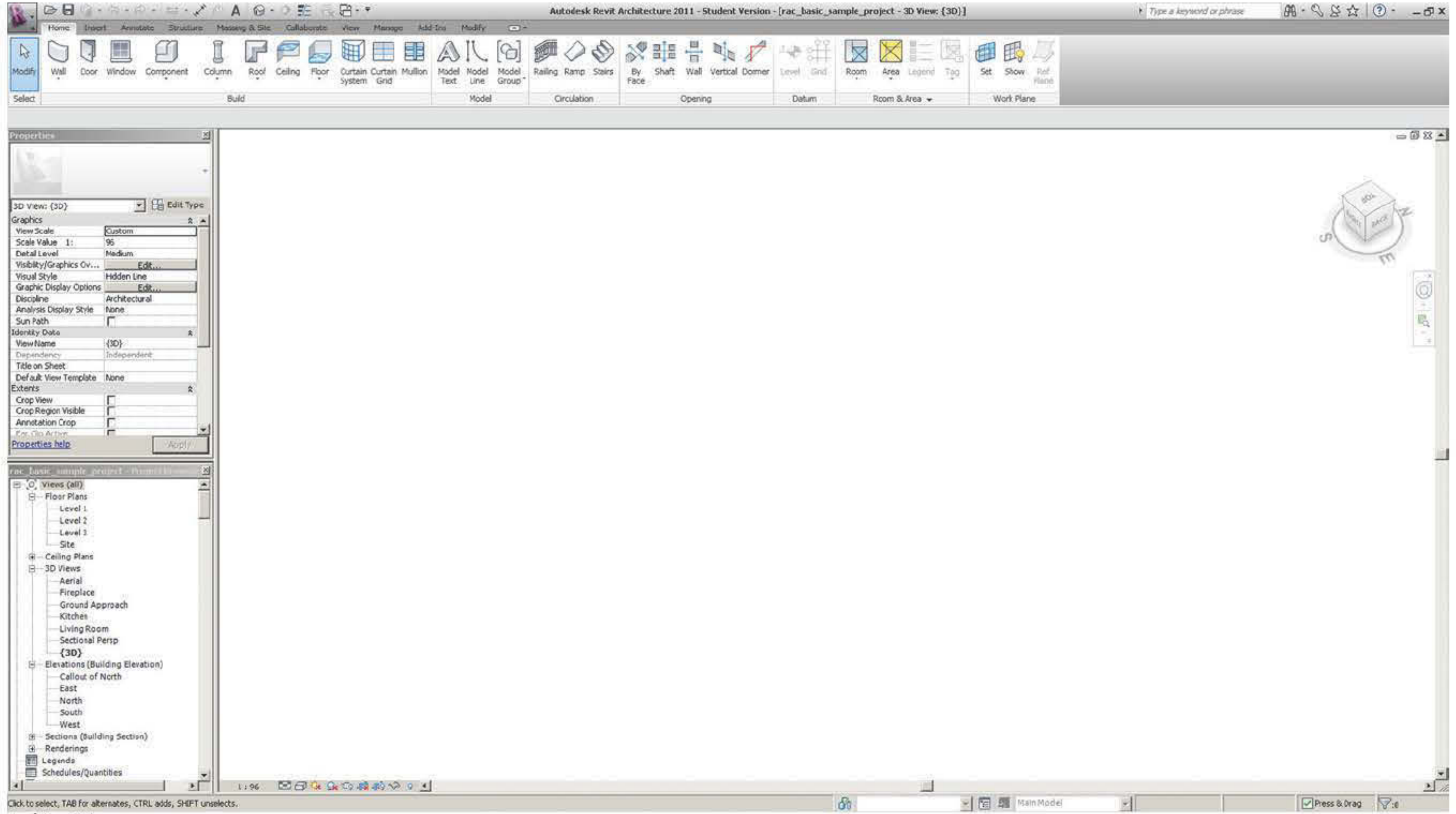
Client/User View



Architect View



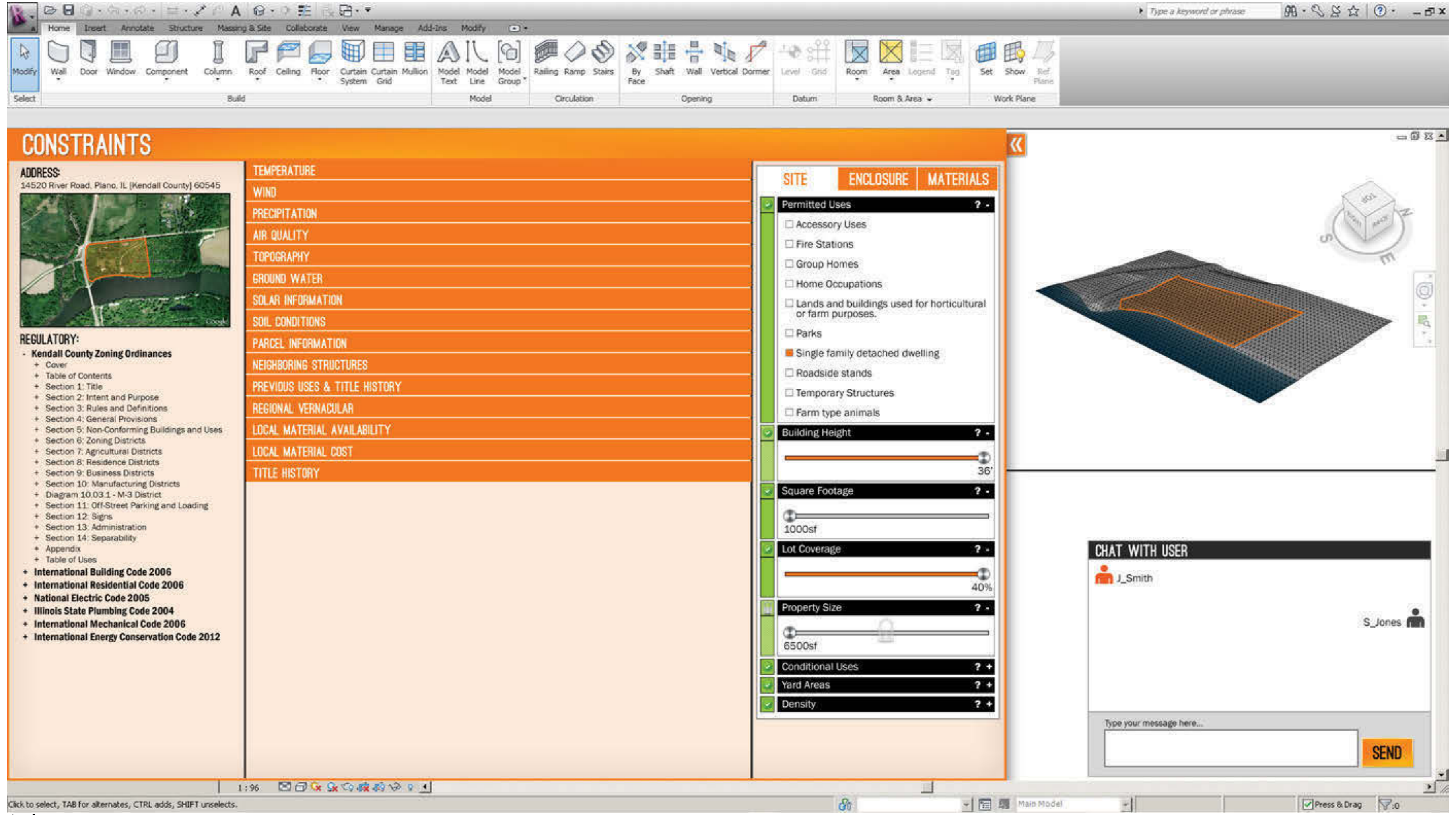
Client/User View



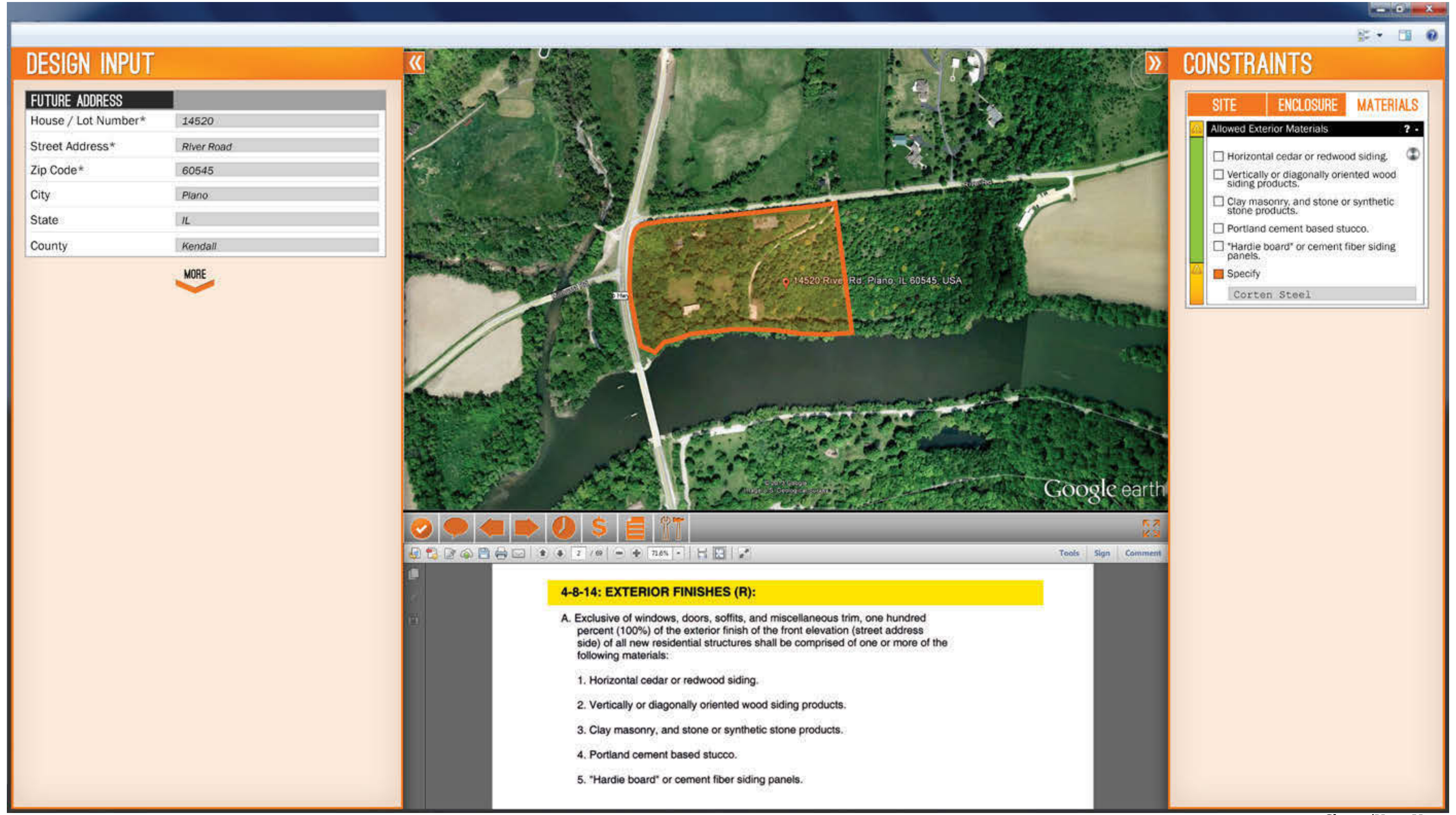
Architect View

wind velocity	solar availability	quality of diffuse light	ultra-violet quantity	ground temperature	parcel dimensions	architectural style of adjacent structures	future land use regulations	feasible treatment of materials	school districts
wind frequency	solar altitude	heating degree days	relative humidity	topography grading	age of adjacent structures	orientation of adjacent structures	contractor requirements	regional vernacular	voting districts
rain quantity	solar azimuth	cooling degree days	vapor pressure	proximity to body of water	current use of adjacent structures and land	proximity of parcel to adjacent structures	real estate transaction laws	smog levels	development tax incentives
rain quality	cloud cover	average high temperature	diurnal swing range	presence of flood zoning	previous use of adjacent structures and land	zoning codes	title history	noise levels	allowable construction hours
snow qunatity	sky illumination value	average low temperature	water table level	general soil conditions	size of adjacent structures and land	building codes	availability of materials	traffic volume	proximity to public safety
solar intensity	quality of direct light	ultra-violet quality	ground water temperature	presence of protected flora	materiality of adjacent structures and land	HOA regulations	cost of materials	crime data	etc.

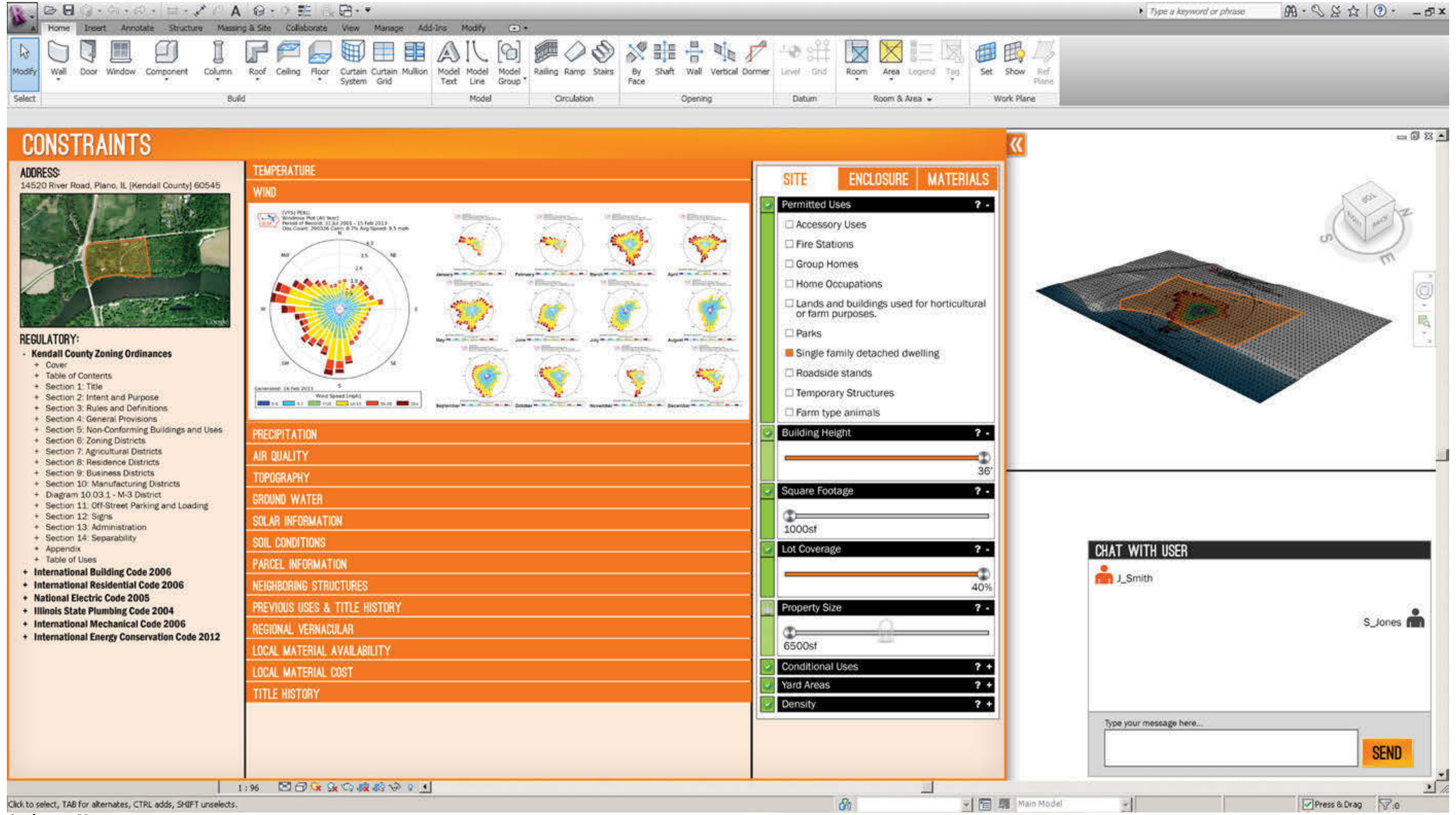
Client/User View



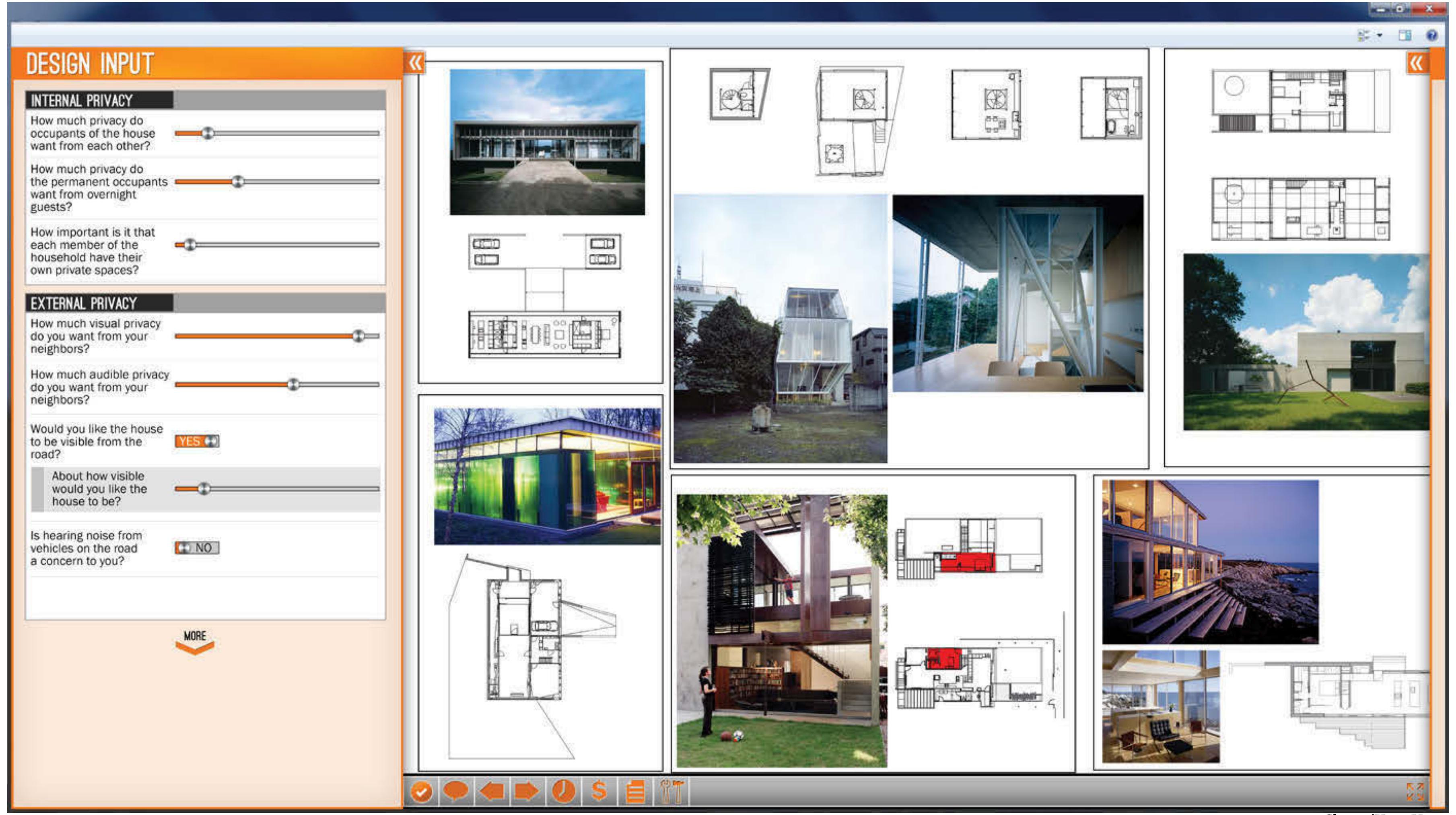
Architect View



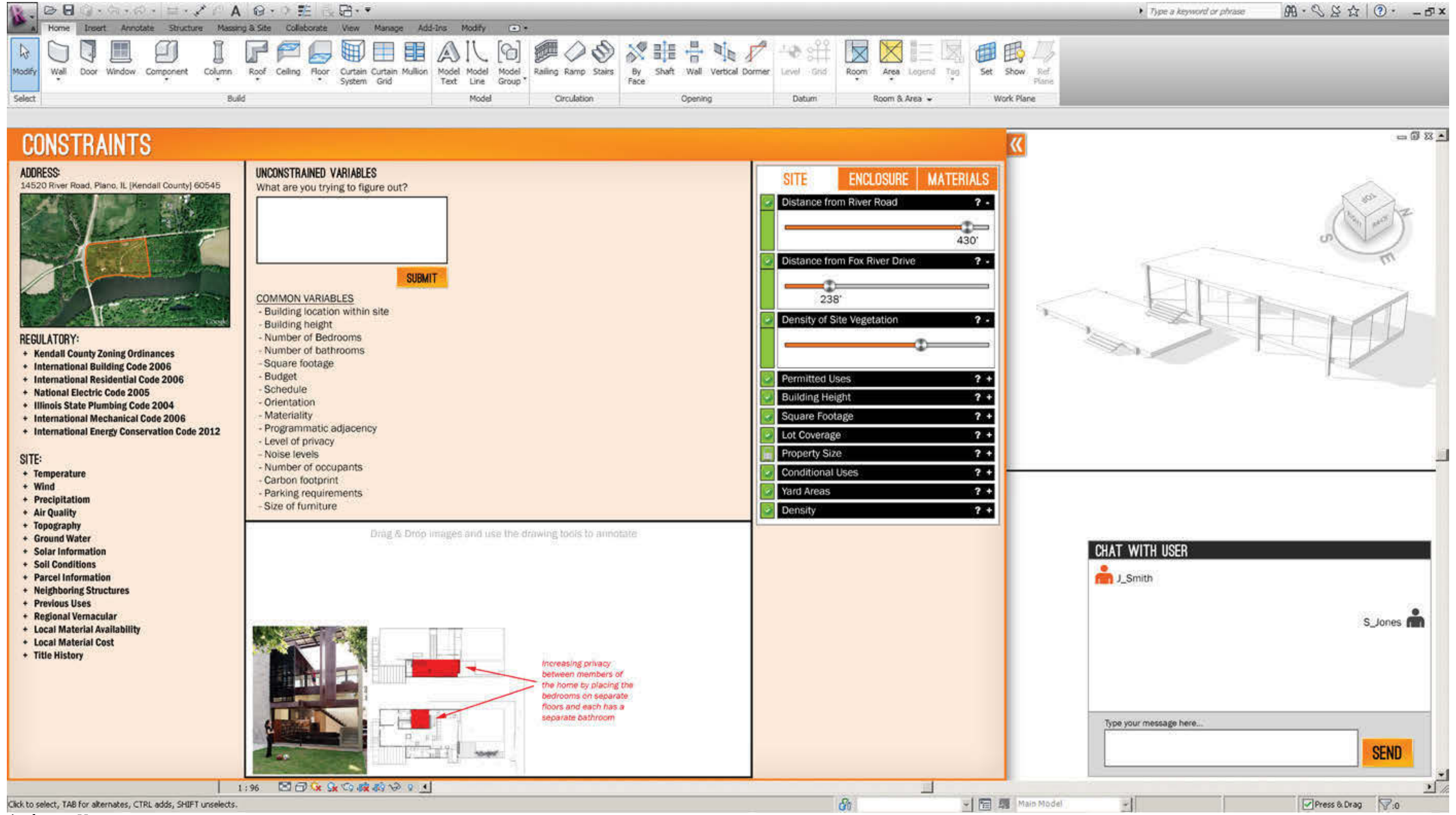
Client/User View



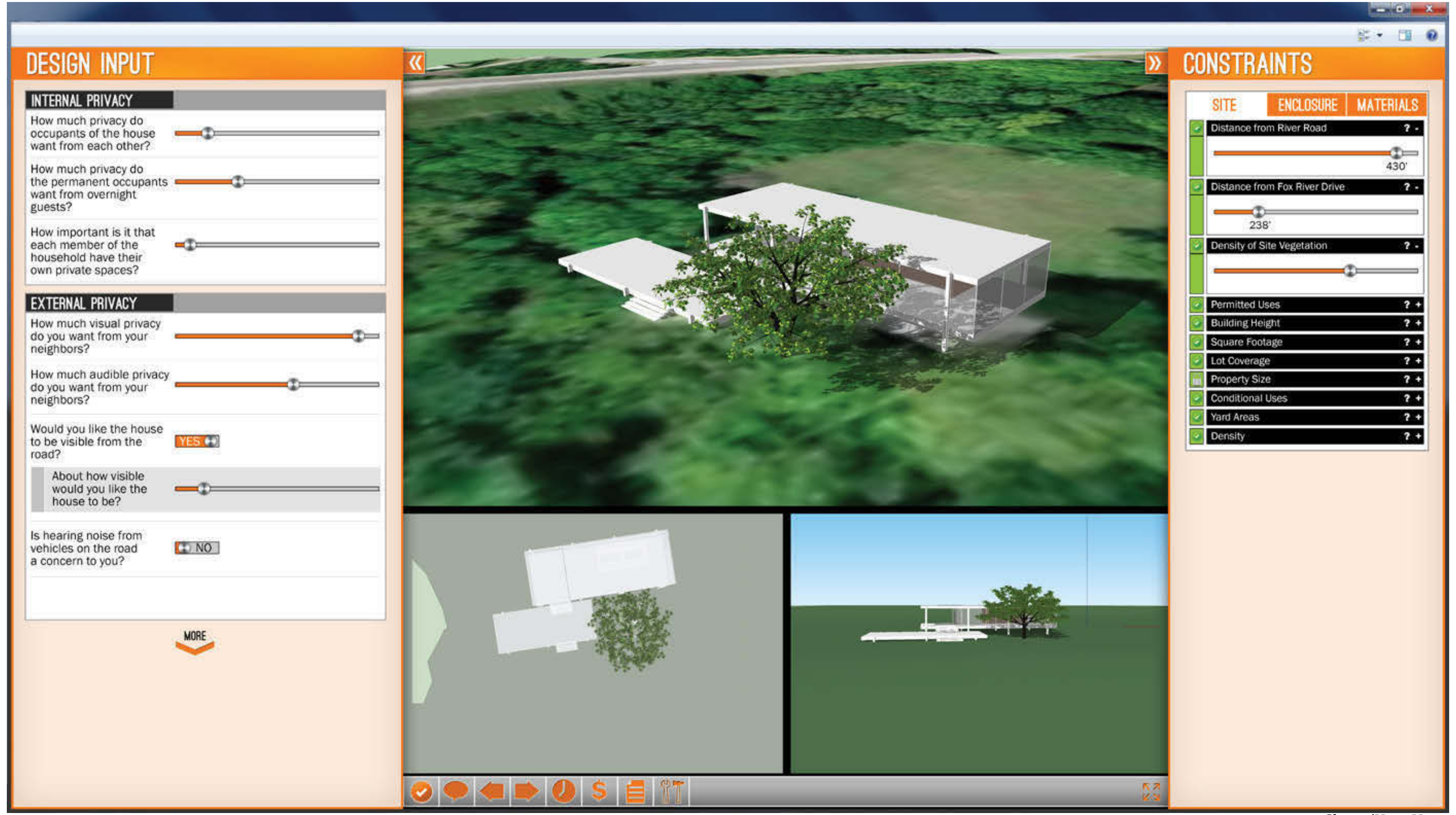
Architect View



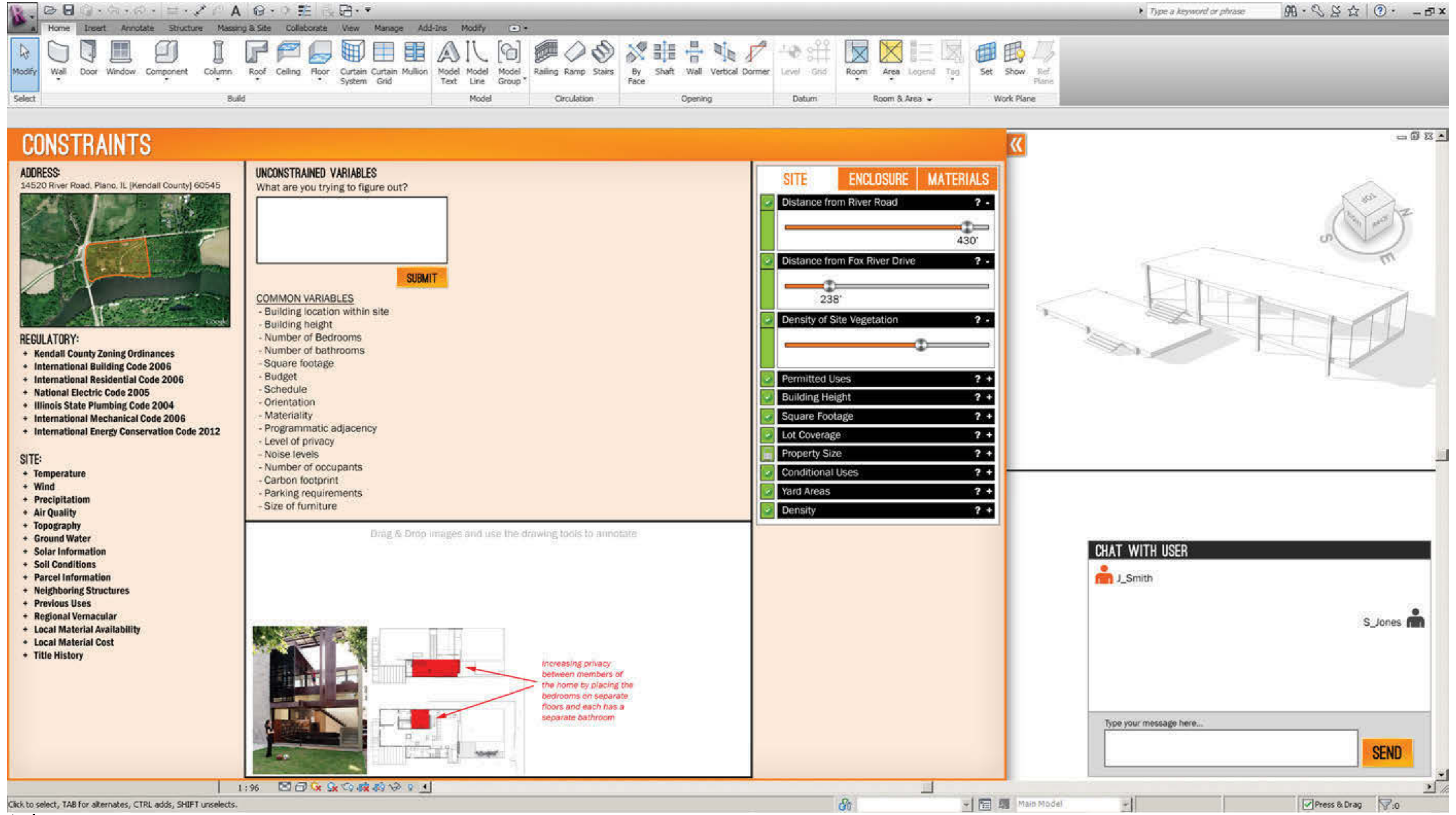
Client/User View



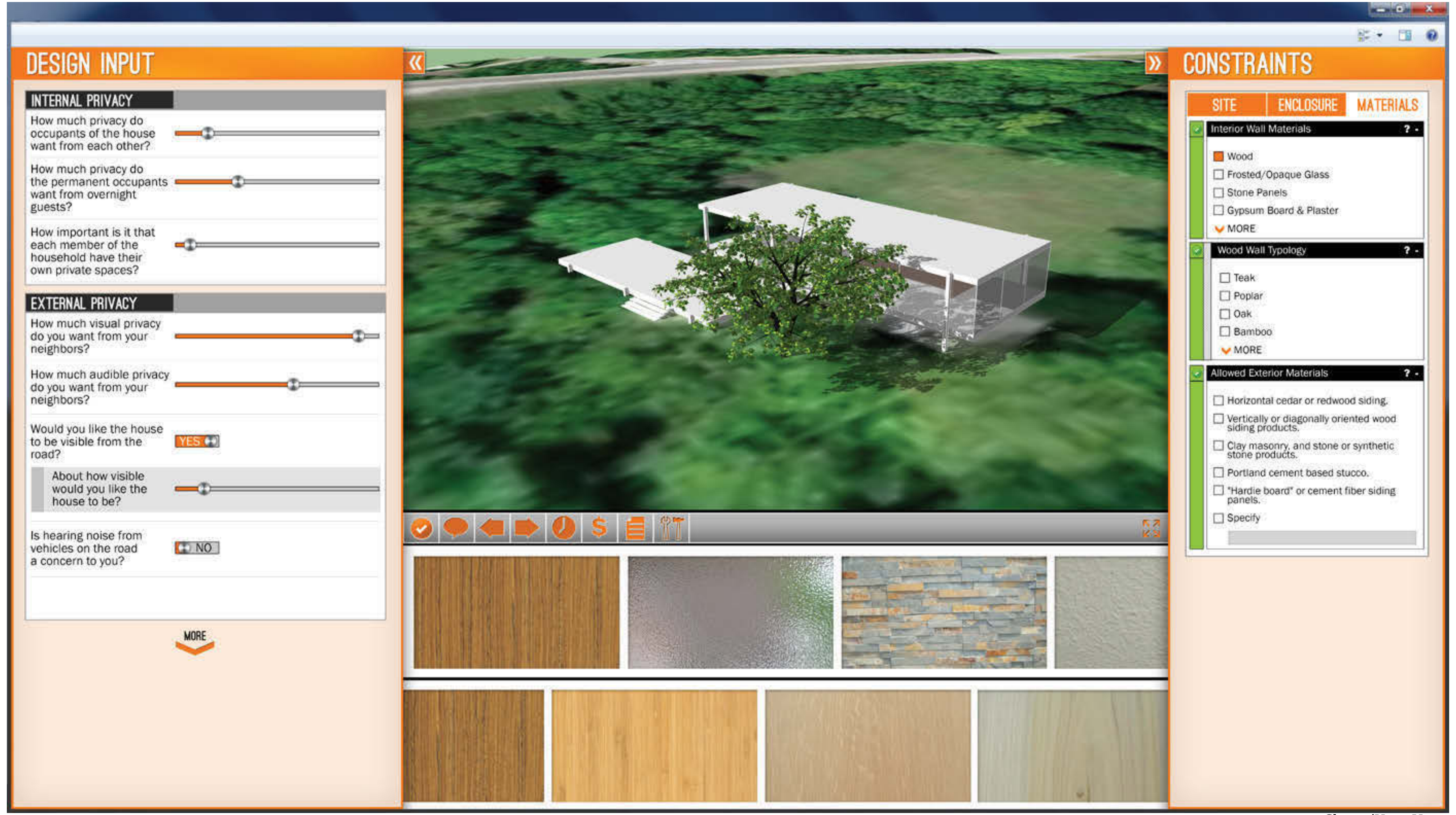
Architect View



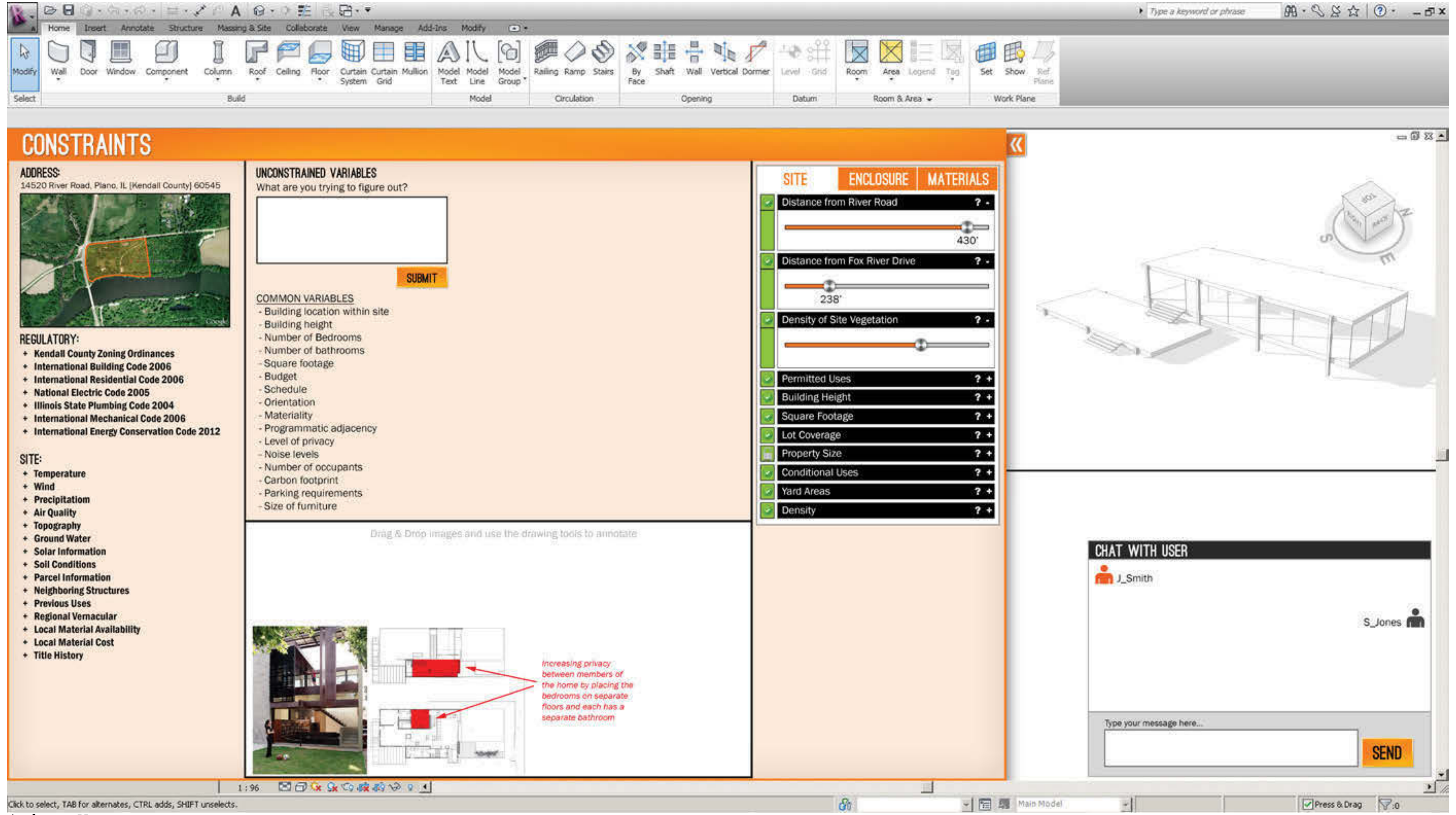
Client/User View



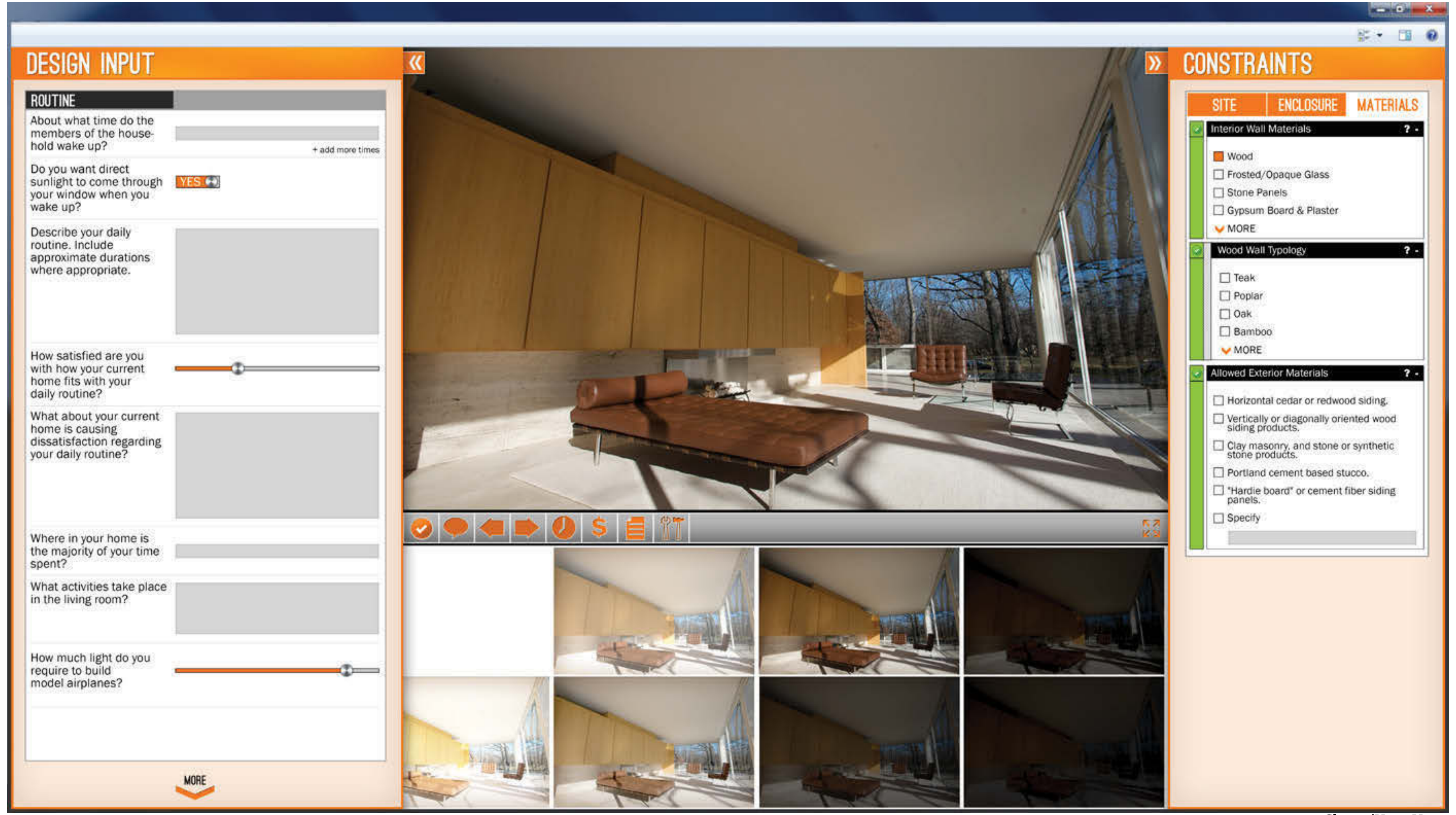
Architect View



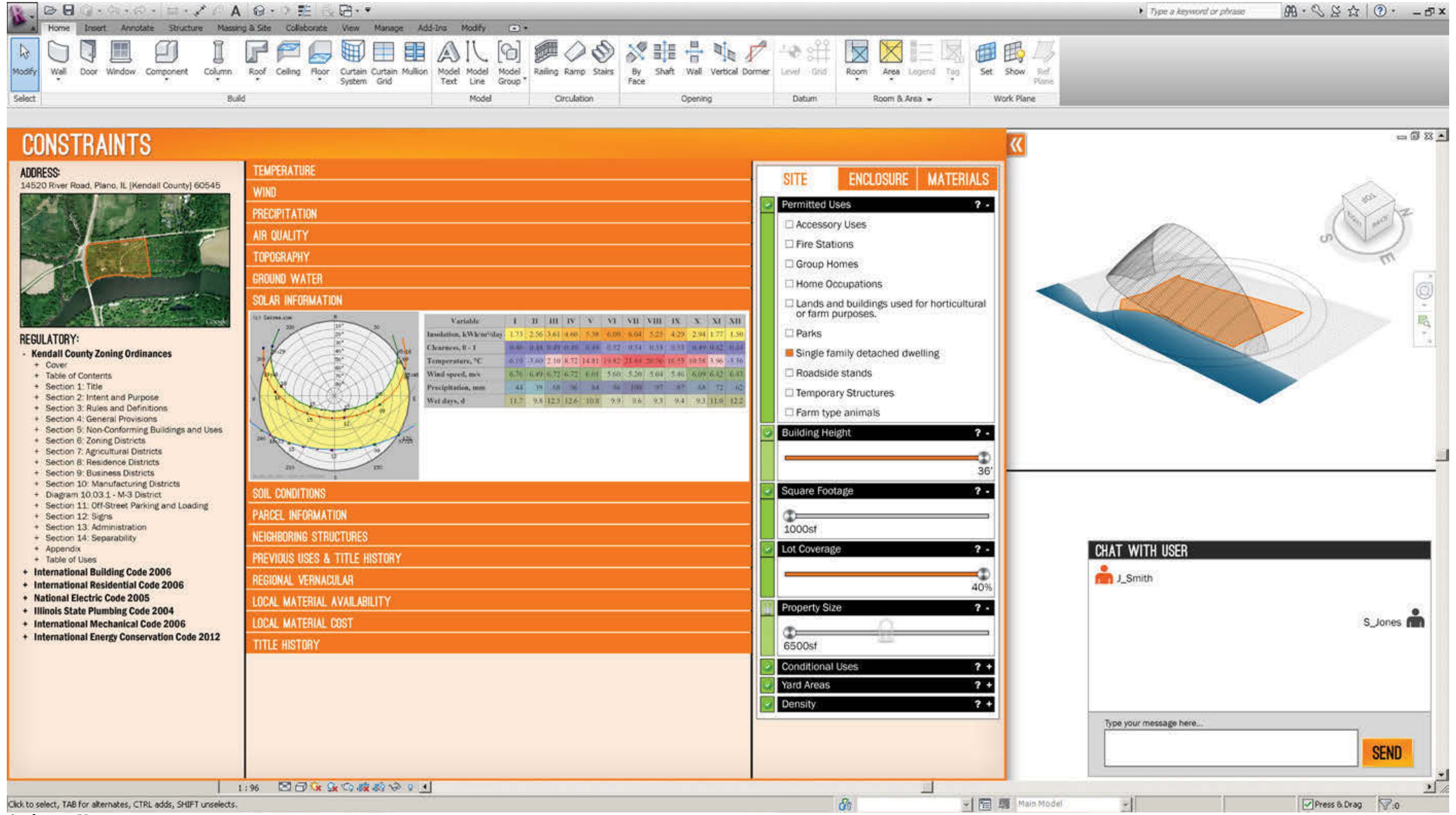
Client/User View



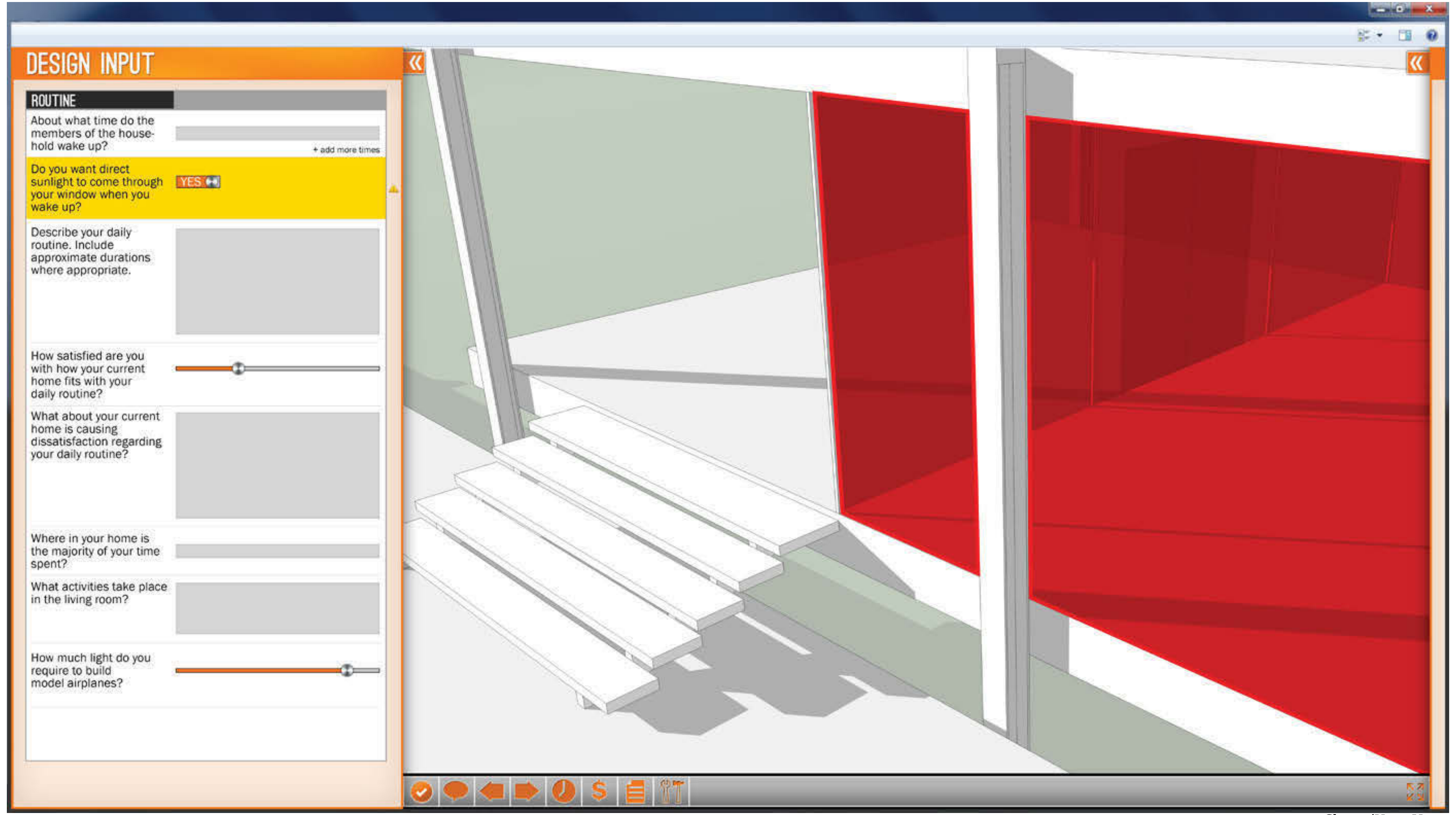
Architect View



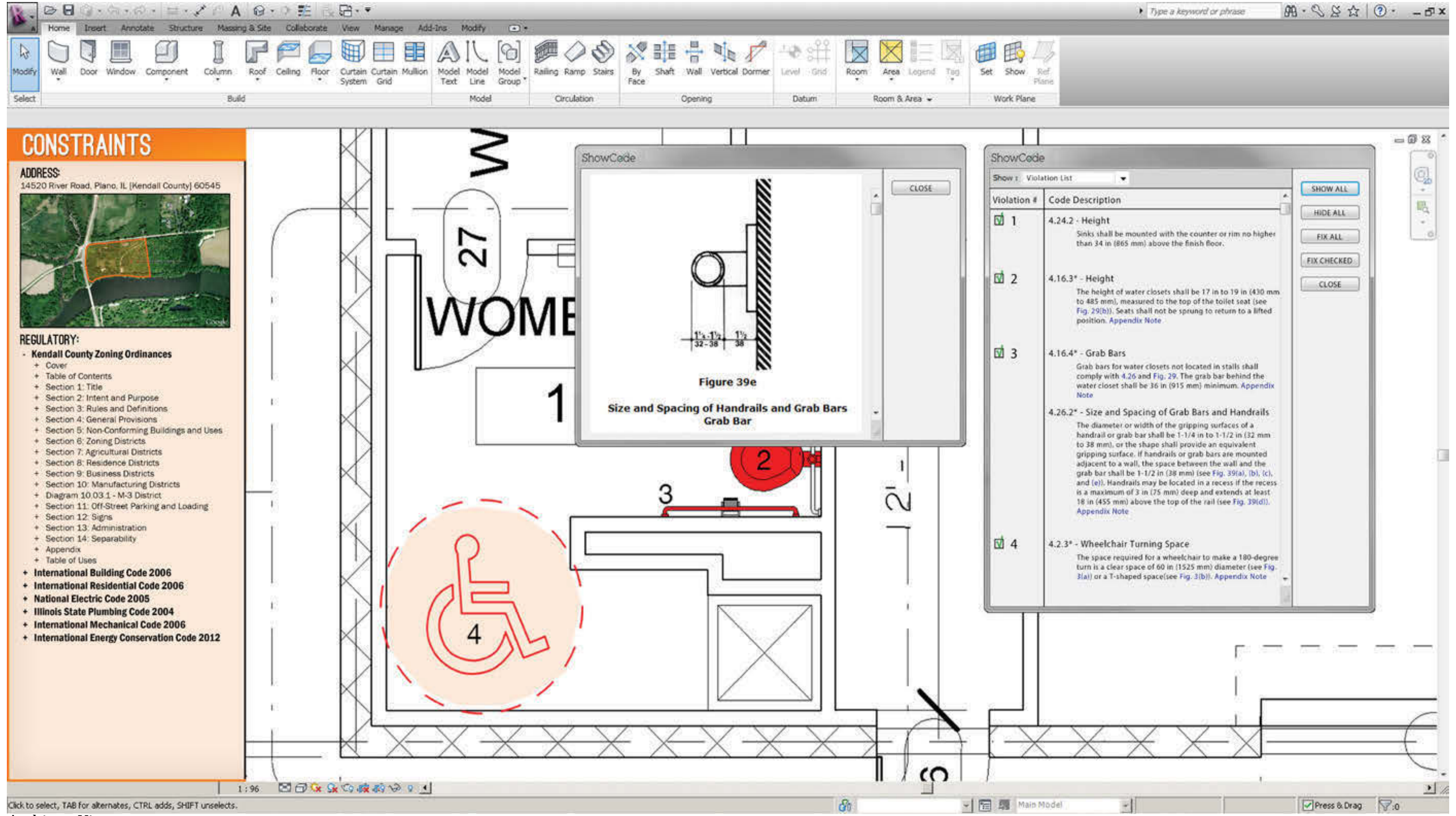
Client/User View



Architect View



Client/User View



Architect View

Continue to work in the software
through completed construction drawings

Download constraints and graphic information
in order to take them to a builder or architect to complete construction drawings

Work without architect support
similiar to typical DIY software while utilizing established constraints

ENDNOTES

1. Nathan Aleskovsky is the founder and CEO of ShowCode LLC – a software company that is developing software application to conduct building code and regulatory compliance analysis.
2. (LaBarre 2008)
3. (U. S. Census Bureau 2013)
4. (LaBarre 2008; U. S. Census Bureau 2013)
5. (LaBarre 2008)
6. (Thompson 2012)
7. (Vardouli)
8. (Friedman 1980)
9. (Vardouli)
10. (Friedman 1980)
11. (Ibid.)
12. (Ibid.)
13. (Ibid.)
14. (Ibid.)
15. (Ibid.)
16. (Alexander, Christopher 1964)
17. (Ibid.)
18. (Ibid.)
19. (Ibid.)
20. (Ibid.)
21. (Ibid.)
22. (Negroponte 1972)
23. (Ibid.)
24. (Ibid.)
25. (Gross 1986)
26. ("Question Wording | Pew Research Center for the People and the Press" 2013)
27. (Lustig and Puget 2001)
28. (Grobman, Yezioro, and Capeluto 2010)
29. (Schoch, Prakasvudhisarn, and Praditsmanont 2011)
30. (Ibid.)
31. (Ibid.)
32. (Ibid.)
33. (Ibid.)
34. (Leatherbarrow 1993)

BIBLIOGRAPHY

Alexander, Christopher. 1964. *Notes on the Synthesis of Form*. London: Oxford University Press.

"Barnes Collection | Tod Williams | Billie Tsien | The New Barnes Shouldn't Work—But Does | By Ada Louise Huxtable - WSJ.com." 2013. Accessed February 5. http://online.wsj.com/article/SB10001424052702304019404577417984288542236.html.

"Chapter 5: Personal Interviews." 2013. Accessed March 25. http://www.fao.org/docrep/W3241E/w3241e06.htm.

City of Plano, Illinois. 2010. "Minimum Construction Standards." http://www.cityofplanoil.com/zoning/MinConstStdsFY10.pdf.

———. 2013. "Welcome to the City of Plano, Illinois." Accessed April 9. http://www.cityofplanoil.com/.

City of Riviera Beach. "CITY OF RIVIERA BEACH, BUILDING DIVISION PERMIT APPLICATION REQUIREMENTS FOR NEW SIGNLE FAMILY OR DUPLEX." http://www.rivierabch.com/filestorage/305/307/1035/1041/permit_app_single_duplex.pdf.

City of Riviera Beach Community Development Department. "Riviera Beach Zoning Map." http://www.rivierabch.com/filestorage/305/307/1035/1039/official_zoning_update_2010.pdf.

"City of Riviera Beach Comprehensive Plan." http://www.rivierabch.com/filestorage/305/307/1035/1039/CRB_Comprehensive_Plan0_(1).pdf.

Colquhoun, Alan. 1996. "Typology and Design Method." In *Theorizing a New Agenda for Architecture*, by Kate Nesbitt, 248–257. New York: Princeton Architectural Press.

"Create House Floor Plans Online with Free Floor Plan Software." 2013. Accessed February 5. http://www.homestyler.com/designer.

Derix, Christian. 2009. "In-Between Architecture Computation." *International Journal of Architectural Computing* 7 (4) (December 1): 565–586. doi:10.1260/1478-0771.7.4.565.

Emrath, Paul. 2009. "NAHB: How Long Buyers Remain in Their Homes." *HousingEconomics.com*. February 11. http://www.nahb.org/generic.aspx?sectionID=734&genericContentID=110770&channelID=311.

Estvez, Alberto T. 2003. *Genetic Architectures = Arquitecturas Geneticas*. Santa Fe: Sites Books.

FAIA, Marvin J. Malecha. 2002. *Reconfiguration in the Study and Practice of Design and Architectureure*. 1st ed. San Francisco, CA: William Stout Publishers.

Fisher, Thomas. 2000. *In the Scheme of Things: Alternative Thinking on the Practice of Architecture*. Minneapolis: University of Minnesota Press.

"Five Basic Types of Questions." 2013. Accessed April 14. http://www4.uwsp.edu/Education/lwilson/learning/quest2.htm.

Frary, Robert. 2013. "A Brief Guide to Questionnaire Development". Virginia Polytechnic Institute and State University. Accessed February 20. http://www.ericae.net/ft/tamu/vpiques3.htm.

Friedman, Yona. 1980. *Toward a Scientific Architecture*. Cambridge, Mass.: MIT Press.

Grobman, Yasha, Abraham Yezioro, and Isaac Capeluto. 2009. "Computer-Based Form Generation in Architectural Design - a Critical Review." *International Journal of Architectural Computing* 7 (4) (December 1): 535–554. doi:10.1260/1478-0771.7.4.535.

———. 2010. "Non-Linear Architectural Design Process." *International Journal of Architectural Computing* 8 (1) (January 1): 41–54. doi:10.1260/1478-0771.8.1.41.

Gross, Mark Donald. 1986. "Design as Exploring Constraints". Thesis Ph. D., Cambridge: Massachusetts Institute of Technology. Dept. of Architecture. http://hdl.handle.net/1721.1/15036.

"Historical Census of Housing Tables - Units in Structure." 2013. Accessed April 24. http://www.census.gov/hhes/www/housing/census/historic/units.html.

"Home & Landscape Design Professional V17 | Punch Software | Official Site." 2013. Accessed February 5. http://www.punchsoftware.com/p-59-home-landscape-design-professional-v17.aspx.

"Home Designing & Decorating Software | HGTV Software." 2013. Accessed February 5. http://www.homedesignsoftware.tv.

Hubbard, Bill. 1996. "The Roots of Architectural Invention: Site, Enclosure, Materials by David Leatherbarrow." *Journal of the Society of Architectural Historians* 55 (2) (June 1): 186–187. doi:10.2307/991121.

"IEM | Custom Wind Roses." 2013. Accessed April 21. http://mesonet.agron.iastate.edu.

Kaspori, Dennis. 2003. "A Communism of Ideas: Towards an Open-source Architectural Practice." *Archis #*.

Kendall, Illinois Planning, Building and Zoning. 2013. "Zoning Ordinance." *Kendall, Illinois Planning, Building and Zoning*. Accessed May 9. http://www.co.kendall.il.us/zoning/zoning_ordinance.htm.

"Kitchen Design Questionnaire." http://www.platinumdesignsllc.com/files/KITCHEN_DESIGN_QUESTIONNAIRE.pdf.

LaBarre, Suzanne. 2008. "Truth in Numbers | Metropolis Magazine." *Metropolis Magazine*. October 15. http://www.metropolismag.com/story/20081015/truth-in-numbers.

Lömker, T.M. 2006. "Solving Revitalization Problems by the Use of a Constraint Programming Language." *17th International Conference on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering*.

Lustig, Irvin, and Jean-Francois Puget. 2001. "Program Does Not Equal Program: Constraint Programming and Its Relationship to Mathematical Programming." *NTERFACES* 31 (December): 29–53.

McLeod, Virginia. 2007. *Detail in Contemporary Residential Architecture*. London: Laurence King Publishing.

Moloney, Jules, and Bharat Dave. 2011. "From Abstraction to Being There: Mixed Reality at the Early Stages of Design." *International Journal of Architectural Computing* 9 (1) (March 1): 1–16. doi:10.1260/1478-0771.9.1.1.

National Trust for Historic Preservation. 2013. "Farnsworth House." Accessed April 16. http://www.farnsworthhouse.org/history.htm.

Negroponte, Nicholas. 1972. *The Architecture Machine: Toward a More Human Environment*. Cambridge, Mass.; London: MIT Press.

———. 1975. *Soft Architecture Machines*. Cambridge, Mass: The MIT Press.

Penttilä, Hannu. 2009. "Services in Digital Design: New Visions for AEC-field Collaboration." *International Journal of Architectural Computing* 7 (3) (September 1): 459–478. doi:10.1260/147807709789621257.

Pérez Gómez, Alberto. 1983. "Architecture and the Crisis of Modern Science." http://hdl.handle.net/2027/heb.05875.

Physician Associates. "Health Questionnaire." http://www.paof.com/sites/default/files/patient-forms/Adult%20New%20Patient%20Forms%20OH.pdf.

Program Evalution and Methodology Department. 1993. "Developing and Using Questionnaires". United States General Accounting Office. http://archive.gao.gov/t2pbat4/150366.pdf.

"Question Order | Pew Research Center for the People and the Press." 2013. Accessed February 13. http://www.people-press.org/methodology/questionnaire-design/question-order/.

"Question Wording | Pew Research Center for the People and the Press." 2013. Accessed February 13. http://www.people-press.org/methodology/questionnaire-design/question-wording/.

"Questionnaire Design, Interviewing and Attitude Measurement - A. N. Oppenheim - Google Books." 2013. Accessed February 13. http://books.google.com/books?id=6V4GnZS7TO4C

Riddle, Bethany Joy. 2008. "Does TurboTax Threaten to Make Accountants Obsolete? Local Experts Say 'no'." *Tri-Cities Area Journal of Business*. January. http://www.tricitiesbusinessnews.com/2008/01/does-turbotax-threaten-to-make-accountants-obsolete-local-experts-say-no/.

Schoch, Martin, Chakguy Prakasvudhisarn, and Apichat Praditsmanont. 2011. "Building-Volume Designs with Optimal Life-Cycle Costs." *International Journal of Architectural Computing* 9 (1) (March 1): 55–76. doi:10.1260/1478-0771.9.1.55.

Thompson, Max. 2012. "It's True: People Don't Know What Architects Do | News | Architects Journal." *Architects Journal*. July 19. http://www.architectsjournal.co.uk/news/daily-news/its-true-people-dont-know-what-architects-do/8633240.article.

Tugend, Alina. 2010. "Too Many Choices: A Problem That Can Paralyze." *The New York Times*, February 26, sec. Your Money. http://www.nytimes.com/2010/02/27/your-money/27shortcuts.html.

U. S. Census Bureau, Demographic Internet Staff. 2013. "Residential Finance Survey Main." Accessed April 24. http://www.census.gov/housing/rfs/.

Vardouli, Theodora. 2012. "Design-for-empowerment-for-design : Computational Structures for Design Democratization". Cambridge: Massachusetts Institute of Technology. http://dspace.mit.edu/handle/1721.1/72864.

———. "'Architecture-by-yourself': Early Studies in Computer-aided Participatory Design". Cambridge: MIT.