Reconsidering the User

Nathan M. Aleskovsky
Syracuse University

Follow this and additional works at: https://surface.syr.edu/architecture_theses

Part of the Architecture Commons, and the Entrepreneurial and Small Business Operations Commons

Recommended Citation
https://surface.syr.edu/architecture_theses/181

This Thesis, Senior is brought to you for free and open access by the School of Architecture Dissertations and Theses at SURFACE. It has been accepted for inclusion in Architecture Senior Theses by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.
RECONSIDERING THE USER

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

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

Founder & CEO, ShowCode LLC
My thesis, Reconsidering The User, is a proposal for a computational architectural design process and the occupant in the design process of a house for transforming how design criteria are gathered and utilized.

Within the scope of the detached single-family home, the thesis argues that a digital process for reconfiguring any, both the architect and the occupant has the potential to create a new paradigm for home design. The methodology I am proposing eliminates some of the primary concerns associated with the way design criteria is gathered from that of a traditional design process and (2) an advancement to the current Do-It-Yourself design solution. This means that of the nearly 60 million single-family homes (figure 6). This is startling given that the desires of either party (figure 5).

Within the context of the detached single-family home, the thesis argues that a digital process for reconfiguring any, both the architect and the occupant has the potential to create a new paradigm for home design. The methodology I am proposing eliminates some of the primary concerns associated with the way design criteria is gathered from that of a traditional design process and (2) an advancement to the current Do-It-Yourself design solution. This means that of the nearly 60 million single-family homes (figure 6). This is startling given that the desires of either party (figure 5).

Why is this important? Currently architecture and the architecture plays a major role in, and is directly responsible for a little over 1 million. The methodology I am proposing eliminates some of the primary concerns associated with the way design criteria is gathered from that of a traditional design process and (2) an advancement to the current Do-It-Yourself design solution. This means that of the nearly 60 million single-family homes (figure 6). This is startling given that the desires of either party (figure 5).

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 design solution. This means that of the nearly 60 million single-family homes (figure 6). This is startling given that the desires of either party (figure 5).
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.

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.
an alternative to the way design criteria is currently gathered

- and -

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

Figure 2. Examples of the software and 3D models used to create home designs.
Figure 1: Diagram of the interactions of architect, client, and
developers in traditional architectural practice.
Economics and efficiency of detached single family home development

Perception of elitism and exclusivity involved in hiring an architect

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

Increase efficiency and reduce cost of design

Create a broader, less privileged relationship

Design control stays with the occupant/user

Direct communication between architects and occupants

Educate the client during the design process
HISTORICAL CONTEXT

In the late 1960s and early 1970s, many individuals did research into user-participatory design and architectural computing. Among those whose work is most noteworthy is evidenced by their continual relevance in contemporary discussions is Nicholas Negroponte, Christopher Alexander, and Yona Friedman. Each of these architects developed methodologies that utilize technology and computing in the analysis of design processes. Each in some sense propose a design process that results in a sample of differentiated projects that are each realized by engaging varying degrees of end-user participation. 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’s work and the respective theories derived from their written work 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 this thesis. Each author is known for the center 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 respective theories and systems and their potential application as such in the realm of design.

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. Each of these architects developed methodologies that utilize technology and computing to the design process, the result of which can be applied to the architectural process.
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.”

Friedman argues that in the past, architecture consisted of a “simple chain of operations” 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 did 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.” 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)”

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 client, architects now design for the specific needs of the average future user. The problem with this approach, he states, is 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 a thousand 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 this bottleneck by implementing a feedback loop. He claims that constructing this new process will, “eliminate information short circuits and therefore unreliability from the message on arrival” , 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 consequences.” Through this process, Friedman recognizes that the future user cannot truly understand the risks involved in making design decisions, stating that it is “essential and dangerous to leave choices to people who have not been properly informed about the consequences of their decisions.” Friedman argues that the role of the architect should be to construct the framework of a building or other programatic space in which the user can participate in 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 own specifications. It is a unique user-centered approach to the planning and programming of programatic spaces in relation to occupant needs.

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).
Like Stockman, Alexander also proposal a systems approach to concepts from biology and engineering. For a basic design problem, the designer must meet requirements that are often incompatible. There are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.

In defining the context and requirements for a design problem, Alexander recommends that designers should seek the presence of underlying rules or principles that can be translated into a number of configurations, processes, or forms which create change. In other words, a 'rule (or the absence of certain design qualities).

Even a designer could find that there are many instances where these requirements, which made sense to start with, are then aggregated in a single model, often within a single model's solution, but not necessarily so. Figure 14 is Alexander's diagram of the complex system of the firm's working process. The variables, represented by the points, are interconnected to one another and not dependent just on data, but also on the firm's organizational structure. Therefore, the diagram also allows for independent adjustments. Here, making is to be seen, in the context of a number of subsystems, as a variable, independent of a single moment of time.
In referring to Negroponte, I will be focusing my analysis on the architect’s work with in the field of computer design. The focus of my project is to emphasize the different designs and their potential to influence the way we interact with technology. In other words, I will be exploring how the design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The first section of the document introduces Negroponte and his work on computer design. It explains how his designs aim to make technology more accessible and user-friendly. The second section discusses the impact of computer design on user experience, highlighting the importance of designing for user needs.

Negroponte’s work on computer design is based on the idea of creating systems that are intuitive and easy to use. He believes that the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

The third section of the document explores the potential of computer design to influence the way we interact with technology. It explains how the design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The fourth section discusses the role of computer design in shaping the future of technology. It explains how the design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.

The fifth section concludes the document by summarizing the key points and emphasizing the importance of designing for user needs. It explains how the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

Negroponte’s work on computer design is based on the idea of creating systems that are intuitive and easy to use. He believes that the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

The potential of computer design to influence the way we interact with technology is immense. The design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The role of computer design in shaping the future of technology is also significant. The design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.

In conclusion, the potential of computer design to influence the way we interact with technology is immense. The design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable. The role of computer design in shaping the future of technology is also significant. The design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.

Negroponte’s work on computer design is based on the idea of creating systems that are intuitive and easy to use. He believes that the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

The potential of computer design to influence the way we interact with technology is immense. The design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The role of computer design in shaping the future of technology is also significant. The design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.

Negroponte’s work on computer design is based on the idea of creating systems that are intuitive and easy to use. He believes that the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

The potential of computer design to influence the way we interact with technology is immense. The design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The role of computer design in shaping the future of technology is also significant. The design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.

Negroponte’s work on computer design is based on the idea of creating systems that are intuitive and easy to use. He believes that the design of a computer should be based on the way people think and interact with technology. This approach requires a deep understanding of human behavior and the way people use technology.

The potential of computer design to influence the way we interact with technology is immense. The design of a computer can affect its usability and the way it is perceived by the user. The design of a computer should be able to enhance the user’s experience and make it more intuitive and enjoyable.

The role of computer design in shaping the future of technology is also significant. The design of a computer can influence the way we interact with technology and how it can shape the way we think about technology.
Looking at these projects the way that I have, I see it very much as a mathematical programming method that is constrained, 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 shoehorn into constraint-based logic.

To elaborate on two terms I just used, mathematical programming and constraint programming. With mathematical programming, a prescribed 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 predetermined sequence of steps is not a mathematical. The design process is a prescribed outline. It is linear and sequential. The sequence of steps is predetermined before beginning the design process.

Constraint programming differs from mathematical programming and is 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, and these may be conditional and non-linear. The collection of constraints indicates the boundaries of the 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.
Constraint Programming

Constraint: 50% of box must be orange

Solutions:
If the constraints are controlling the limits of doing all its important, for the information, we must be 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 home. However, rather than information gathering, most of the information is obtained through personal interviews. 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 them credibility and limits embarrassment in the eyes of onlookers, rather than giving a truthful reply.

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.
The BVO model employs constraint programming (CP) instead of mathematical programming as the declarative optimization technique. This is in accordance with the goals of the model, which are stated as a sequence of feasible and optimal solutions. It is a Constraint Satisfaction Problem (CSP), to be solved by a Constraint Satisfaction Solver (CSS).

In 2010, Yaniv Gutfreund published an article in the International journal of Architectural Computing entitled “BVO Architectural Design Framework: A constraint-based approach for architectural design.” Its main purpose is to present a methodology to generate feasible solutions for a design problem. The methodology is based on an optimization process.

The paper introduces a new method for architectural design that is based on constraint programming. The method is characterized by its ability to define the problem at a high level and to automatically generate feasible solutions. The methodology is based on a constraint satisfaction approach that is analogous to the way a human architect would approach a design problem.

The methodology is characterized by its ability to define the problem at a high level and to automatically generate feasible solutions. The methodology is based on a constraint satisfaction approach that is analogous to the way a human architect would approach a design problem.

In developing his program, Schoch relied on various design constraints that describe how a design problem could be formulated. For example, Schoch used the optimization process to determine the best combination of design parameters.

The methodology is characterized by its ability to define the problem at a high level and to automatically generate feasible solutions. The methodology is based on a constraint satisfaction approach that is analogous to the way a human architect would approach a design problem.

In developing his program, Schoch relied on various design constraints that describe how a design problem could be formulated. For example, Schoch used the optimization process to determine the best combination of design parameters.
Architectural design is affected by rules.

Rules can be used to constrain the solution space.

Computer can extend the number of feasible options.

Design solution has to meet specific requirements (Constraints).

The design has to strive for specific goals (Objectives).

There are choices available that meet the constraints and 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

Minimize: $LCC_{pv} = CC_{pv} + EC_{pv} + OMC_{pv} + RRC_{pv}$
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).

Figure 3: User interface, initial setup and generated alternatives catalogue. Building's initial form and the possible use of the selected envelope.
 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 must work out site, enclosure, and materials before considerations of style and aesthetic can be applied.

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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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 such as 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.

The interface design 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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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 design 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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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 design 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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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 design 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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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 design 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 that the user is typically unaware of the full scope of their design decisions. This lack of awareness begins with the moment the user enters building code violations or unique design features that hinder proper cooling. When that occurs, one can imagine all of the potential design scenarios that are possible for the user based on the decisions they have made.

The primary objective of the interface is to collect necessary and accurate data by translating the needs of the architect into questions that are understandable to the user. This is necessary because the reason for asking a question can be very different between an architect and an occupant. A client may ask, “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 windows 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.
<table>
<thead>
<tr>
<th>Environmental Factors</th>
<th>Architectural and Planning Factors</th>
<th>Legal and Economic Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wind velocity</td>
<td>Solar availability</td>
<td>Future land use regulations</td>
</tr>
<tr>
<td>Wind frequency</td>
<td>Solar availability</td>
<td>Feasible treatment of materials</td>
</tr>
<tr>
<td>Rain quantity</td>
<td>Solar altitude</td>
<td>School districts</td>
</tr>
<tr>
<td>Rain quality</td>
<td>Solar azimuth</td>
<td>Architectural style of adjacent structures</td>
</tr>
<tr>
<td>Snow quantity</td>
<td>Solar altitude</td>
<td>Age of adjacent structures</td>
</tr>
<tr>
<td>Solar intensity</td>
<td>Solar altitude</td>
<td>Orientation of adjacent structures</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Contractor requirements</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Real estate transaction laws</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Smog levels</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Development tax incentives</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Smoke levels</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Noise levels</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Allowable construction hours</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Proximity to parcel to adjacent structures</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Zoning codes</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Title history</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Building codes</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Availability of materials</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Traffic volume</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Proximity to public safety</td>
</tr>
<tr>
<td>Desert intensity</td>
<td>Solar altitude</td>
<td>Etc.</td>
</tr>
</tbody>
</table>

Legend:
- Environmental Factors: Physical characteristics of the environment.
- Architectural and Planning Factors: Factors related to the design and planning of buildings and structures.
- Legal and Economic Factors: Legal and economic considerations relevant to the project.
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 similar to typical DIY software while utilizing established constraints.
1. Nathan Aleskovsky is the founder and CEO of ShowCode LLC – a software company that develops software applications 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)


27. (Lustig and Puget 2001)

28. (Grobman, Yezioro, and Capelutto 2010)

29. (Schoch, Prakasvudhisarn, and Praditsmanont 2011)

30. (Ibid.)

31. (Ibid.)

32. (Ibid.)

33. (Ibid.)

34. (Leatherbarrow 1993)