Usability principles and best practices for the user interface design of complex 3D architectural design and engineering tools

Ghang Lee\textsuperscript{a,},* Charles M. Eastman\textsuperscript{b}, Tarang Taunk\textsuperscript{b}, Chun-Heng Ho\textsuperscript{c}

\textsuperscript{a}Department of Architectural Engineering, Yonsei University, 134 Sinchon-Dong, Seodaemun-Gu, Seoul 120-749, Republic of Korea
\textsuperscript{b}Georgia Institute of Technology Atlanta, GA, USA
\textsuperscript{c}National Cheng Kung University, Taiwan

Received 8 January 2008; received in revised form 23 July 2009; accepted 7 October 2009
Communicated by E. Motta
Available online 13 October 2009

Abstract

This study proposes usability principles for the user interface (UI) design of complex 3D parametric architectural design and engineering tools. Numerous usability principles have been developed for generic desktop or web applications. The authors tried to apply existing usability principles as guidelines for evaluating complex 3D design and engineering applications. However, the principles were too generic and high-level to be useful as design or evaluation guidelines. The authors, all with more than 10 or 30 years of experience with various CAD systems, selected and reviewed 10 state-of-the-art 3D parametric design and engineering applications and captured what they thought were best practices, as screenshots and videos. The collected best practices were reviewed through a series of discussion sessions. During the discussion sessions, UI design principles underlying the collected best practices were characterized in the line of existing UI principles. Based on the best practices and the derived common UI principles, a new set of refined and detailed UI principles were proposed for improving and evaluating 3D parametric engineering design tools in the future.

\& 2009 Elsevier Ltd. All rights reserved.

Keywords: User interface (UI); Computer-aided design (CAD); 3D engineering design; Best practices; Usability principles

1. Introduction

Design and engineering are the base activities that convert raw material or technologies into useful products. In the last two decades, most design and engineering work has become computerized. While previously, engineers worked on drafting tables and used slide rules, they now use computer-based applications to lay out and analyze their designs. Today, we can assume that somewhere around one million US workers spend a significant part of their working day in front of computer-aided engineering or design tools. (Engineers comprise over 1.4 million jobs, drafters in all fields are another 0.25 million; architects are another 0.13 million (US Bureau of Labor Statistics, 2004).)

Continuous evolution of these tools has increased their power tremendously. Started about 35 years ago as 2D drawing tools, they have evolved into highly specialized parametric 3D modeling design tools, interfaced with supporting analysis and data extraction applications. The applications have become domain-specific with objects, properties and relations incorporating knowledge in specific engineering domains, for example, for steel detailing and fabrication, piping systems, precast concrete design and fabrication, and mechanical system design and layout. These applications often are applied to designs with more than a hundred thousand objects. They support further embedding of design and compositional rules for company-specific knowledge. Interfaces to external applications for engineering analysis and simulation are provided to predict the product’s future fabrication, performance and operation. By computer-enhancement of these tasks and developing new technologies that can only be realized computationally, the processes of design,
engineering, and production planning has allowed new products that could not be produced using manual processes, in a timeframe and with performances that could be not realized previously.

An indication of the potential level of effort involved in the setup of parametric modeling applications for production use in the aerospace domain. Boeing spent over one billion dollars to define the rule-bases and interfaces in the parametric modeling environment to be used in the design and production of the 777 line of airplanes (Boeing, 1997). This was not a re-design of the platform system, but rather using existing UI and modeling tools to capture the rules and business practices that Boeing expected to utilize in the product’s development and manufacture.

However, the functionality offered by these software systems has come at a cost; the tools themselves have become highly complex. Users are expected to apply their domain knowledge both collaboratively with other engineers and also with the software to embed and apply the engineering analysis expertise and production rules to successfully and efficiently accomplish their work. Users without in-depth knowledge of the domain are not able to use the tools effectively. In addition, the domain knowledge incorporated is quite extensive, dealing with complex geometries, material properties, engineering and production practices, as well as business practices regarding communications, coordination, review processes and the detailed data requirements associated with each.

Design and engineering software applications include several hundred operations, many of which include submenus and detailed options. Many applications support multiple representations of the design, such as the physical representation, more abstract structural or thermal element-based representations, and sometimes process representations dealing with sequencing and scheduling. The organization of this broad set of functionality is highly varied within different software products. As a result, learning how to apply the appropriate domain knowledge in carrying out the design and engineering tasks must address each of the different representations and tasks. Furthermore, the software offerings are young and quickly evolving – new releases are often generated quarterly and new functional capabilities are added with each release. These additions are done accumulatively and often by users’ requests. In this environment, new functionality is sometimes squeezed into an existing user interface structure without having enough time to consider a long-term usability framework.

Beginner software training classes typically involve a week of instruction and weeklong advanced classes are typically required for advanced topics. In practice, efficient use of such complex tools requires continuous use; access to sequences of operations and remembering needed contextual conditions cannot be retained without constant use. These difficulties has led to a new position in many industries, called a modeler, which we found is a common employment position in many firms that have adopted 3D parametric modeling applications. The responsibility of a modeler is to interpret and convert engineering intentions into a model using a 3D parametric design application. An obvious future bottleneck when more design and engineering processes are automated is this split between domain expert and tool expert. It is important for engineers with domain expert knowledge to be able to directly use such systems, from both work productivity and legal liability points of view. In such circumstances, it becomes crucial to have a consistent interface that allows engineering users to efficiently learn and use the design and engineering system.

Using surveys and our own review, we collected 627 issue reports that were distilled into 179 UI issues regarding a complex building design and engineering system, in order to understand the specific and in-depth UI problems. We then reviewed 10 3D design and engineering applications for their UI design and used these to compile best practices for different user interface types. The emergent best practices were compared with existing UI principles and guidelines found in prior published work (Burns and Hajdukiewicz, 2004; Galitz, 2007). However, we found that these were mostly generated for informing the design of generic desktop applications, such as text editors, or web applications, and were too generic for highly visual, complex 3D parametric modeling tools. Based on the usability analysis results, the collected best practices, and our experience as developers of a complex design and engineering system, we propose a set of detailed and refined UI principles for complex design and engineering systems. In this paper, we use the terms “usability principles” and “UI principles” interchangeably. We refer to usability principles as practical guidelines or heuristic checklists for designing user interfaces.

2. UI principles and guidelines

As the field of UI interface design has matured, the principles of user interaction have evolved also, from a single “universal” set into a more complex but still loosely defined taxonomic structure. Some principles are oriented toward guidelines for generation of all user interfaces, while others are representation-specific, for example, Web pages (Nielsen, 2000; Quesenbery) or databases. Some authors consider the universal principles to be function-free and can be interpreted and mapped to the more detailed representations. Norman (1990), in his book *The Design of Everyday Things*, suggests that good visibility of the perceived properties of an object, ability to construct conceptual models from it, and providing efficient feedback to users’ actions are the overall principles of good design for usability. We found Nielsen and Molich’s 10 general rules of thumb or heuristics for user interface design (Molich and Nielsen, 1990), were good for evaluating systems or design prototypes of basic desktop or web applications, but their scope did not extend to cover the UI design of complex 3D modeling tasks that we are concerned with. Similarly, the *Principles to Support*
Usability suggested by Dix et al. (1997) in their book *Human–Computer Interaction* and the First Principles of Interaction Design proposed by Tognazzini (n.d.), were not sufficient in terms of informing the design of UI for tasks specific to 3D modeling. While many efforts have been directed to the general principles of interface design, it is at the more detailed and applied area that task-specific interfaces apply, which is the focus of the work presented here.

It is not our intention to develop a general taxonomy of UI principles, but a general structure can be conceived. Some proposed principles are meant to be general and apply to all types of UIs. More detailed levels of UI principles can be broken out in different ways. In our consideration, it seems appropriate to break down UI issues according to the different representations of information presented and its creation, manipulation, structuring and elaboration. In this type of taxonomy, spreadsheets and databases are a representation involving highly structured numerical content but usually mapped through various screen-level forms for input, editing and manipulation (Jagadish et al. 2007). Two-dimensional drawing applications, based on raster or vector representations are a second representation type with their own UI practices (Vdraft, n.d.). Our interest is in custom-defined and structured 3D object modelers for use in the area of architecture, engineering, and construction, a third functional area with special UI requirements and principles. We refer to these in the paper as architecture and engineering (AE) systems.

Our applications of interest involve multiple representations. In some cases an application or task has been conceived as having a primary representation, which provides the cross-referencing base for all other data representations, while in other cases multiple representations are at an equal standing; in each case linking and cross-referencing, where consistency management between representation becomes an important model management concern. The essential character of the parametric 3D modeling systems addressed here is that the base reference for accessing, editing and navigating the representation is through virtual 3D Cartesian space and complexly configured object shapes. It is broadly recognized that engineering objects are not inherently shapes but more abstract functional objects that are assigned shape attributes in their realization (Csabai et al., 2002; Eastman et al., 2004): some tools recognize this and are more fully represented in such tools as enterprise resource planning (ERP) systems, implemented as a database repository. Still, in these systems, the parametric modeling tools are a primary authoring tool for engineering design. Significant amounts of textual and numerical properties are associated with the space objects in the primary representation, but in the tools we are considering these are typically managed through the 3D tools.

The UI problems of complex design and engineering systems can be interpreted from a cognitive perspective. The complexity and range of functionality of the system requires a very long time to first learn, then requires continued practice, to move from conscious attention to a more pre-conscious automatization (Sasse, 1997) and then to maintain the automatization. The complexity of operations, often requiring many sequential actions, each with many parameters, imposes a cognitive load requiring high degrees of concentration. The attention required by the software system to operate it competes with the attention associated with domain knowledge that is required to carry out the task. Such conflicts between tool complexity and the tasks to be realized through the tool have been recognized elsewhere, particularly in educational software (Sweller, 1988).

The perceived need we are addressing can be summarized as follows:

- Some of the existing UI principles were not task- or representation-specific to the 3D parametric modeling domain. Some came close to matching some problem examples, but the correspondence of issues and response were partial at best. This often occurred due to the ambiguity of the criteria definition. In such cases, we renamed or redefined the criteria to better explain the desired design phenomena.

- Many UI design examples of 3D engineering design systems that were task or domain-specific simply did not match with any of the existing UI principles.

In order to overcome these limitations, we propose a set of UI principles for 3D AE systems, defined by understanding major UI issues that users of a complex AE system suffer and then by reviewing best practices of 10 state-of-the-art 3D computer-aided design systems in related fields.

3. UI issues of a 3D AE system

User interaction design aspects are typically task-specific. Particularly in our case, user interaction involves navigation in very large models, 3D selection and input (which involves special consideration because the cursor in 3D depicts a path from the screen into the 3D space and not a single point), and management of visually rich content, which are not necessarily relevant in many other domains.

Specific UI issues of a complex AE system were collected through an on-line bug-report system and questionnaires used by trainees that participated in the training sessions of the system. The questionnaire was composed of 32 open-end questions such as “What operations according to you were the most difficult to perform? Why did you think they were difficult?” They were also collected through focus group interviews with system trainers from October 2004 to May 2005. The aim of the survey was to understand the specific issues and problems faced by novice users of the system. Since the issues and problems were collected mostly
through the on-line bug-report system and we did not collect users’ information to encourage them to participate in the survey, the number of respondents is not known. A total of 627 UI issue items were collected. These collected UI items were grouped as 179 distinct UI issues after eliminating redundant items. They were then categorized by topic, criticality, and complexity as shown in Fig. 1. The issues were initially classified by the users who generated the bug reports. There were many unclassified issues in the initial set. The initial classification was reviewed and filled in by the authors through multiple iterations to verify consistency of classification.

The topics were issues related to: 1. dialog boxes, 2. drawings and reports, 3. the help function, 4. modeling, 5. domain terminology, 6. general system terminology, 7. view and navigation, 8. the menubar, 9. the toolbar, and 10. others.

The criticality of an issue was determined by two factors:

- **Usability**: Is this problem harmful for the functionality of the system or preventing users from using a certain function in an intended way?
- **Frequency**: Has this problem been repeatedly identified and reported by users?

The complexity issue required in-depth investigation whereas a simple issue is an issue that had an obvious solution and may be fixed immediately. Examples include typos or missing items. The complexity of an issue was determined by three factors:

- **Impact on current users**: Will the resolution to this problem produce a large (negative) impact on current users of the system?
- **Difficulty of validation**: Is it difficult to validate/evaluate the proposed solution?
- **Ambiguity of solution**: Is the proposed solution unclear or absent?

The most problematic issue topics of the system were the modeling issue (26%, 46 distinct items), the dialog box issue (25%, 45 distinct items), and the domain terminology issue (17%, 31 distinct items) (Fig. 2 and Table 1). The critical value represents the urgency of an issue and the complexity value represents the difficulty of solving the issue. Thus, from a developer point of view, it is logical to solve problems with higher criticality values and lower complexity values first. We named the index **improvement priority index**. In this study, the criticality index and the

<table>
<thead>
<tr>
<th>Subject</th>
<th>Issues &amp; Suggestions</th>
<th>Relevant Category</th>
<th>Critical</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value for &quot;Factor of bolt edge distance&quot; (TS 11.1b6)</td>
<td>Setup -&gt; Options -&gt; Preferences tab The default value for &quot;Factor of bolt edge distance&quot; is 1.2. However, the commonly used bolt edge distance is 1.25 and not 1.2. Because the field allows only one decimal place, if one types 1.25 in the field, it automatically gets rounded off to 1.3. During training we were told that although in the dialogbox the value appears to be 1.2, the system actually reads it as 1.25. This is highly deceiving to the user. <strong>Suggestion:</strong> Allow for more than 1 decimal places. Change the default &quot;Factor of bolt edge distance&quot; as 1.25.</td>
<td>Y</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. An example of UI issues.

Fig. 2. UI Issues by Topic (total of 179 distinct UI Items collected).
complexity index were evaluated as either Yes (=1 point) or No (=0 point). But ideally, the criticality index of an UI issue can be evaluated by users on a 1–5 scale and the complexity index by system developers again on a 1–5 scale. We calculated the criticality and the complexity of each topic area by dividing the number of critical issues by the number of UI issues and the improvement-priority index for each topic area by dividing the criticality value for each area by the complexity value for each area. Table 1 shows the analysis results for each topic area for the specific application that we examined in this UI issue survey. This improvement priority values are only meaningful to a specific software system and the broader goal of this survey was not to determine which issue must be improved first, but to understand and identify general UI issues of complex CAD systems. Thus this analysis result is presented only as an example of how the captured criticality and complexity values can be used in determining the improvement priority of UI issues in practice.

As the next step, we reviewed interfaces of 10 state-of-the-art computer-aided design systems to find solutions to improve issues in these areas and tried to generalize the findings as a set of UI principles for AE systems.

4. Best practices

User interaction is a dynamic area where available technology is exploited to facilitate man–machine communication. Initially, this grew out of the base capabilities of graphical UIs and bitmap operations. Controllable bit-maps for cursors, support for cursor pass-over pop-up tips, and real-time dragging of objects in a window, are examples where hardware has been exploited to provide enhanced interaction. These are hardly based on general principles today, but rather represent a set of effective “tricks” or “best practices” that people have invented to simplify and facilitate complex conditions.

Working on three-dimensional layouts through a two-dimensional screen is one of the classic user interface issues in modern engineering and design tools (Foley et al., 1991). The issue of extracting a third dimension from essentially 2D cursor input has always been a challenge. Methods to lay out parametric objects and manipulate them in 3D space have evolved over time as a collection of “best practices”. Certain methods of interaction for the different 3D layout tasks have been found to be particularly effective. Only occasionally are new methods of interaction developed. Usually they show up in a particular product and competing products usually adopt these within a succeeding generation of software development. In our domain, UI solutions such as separate sketching tools for creating parametric section profiles of objects, semi-transparent objects for seeing through complex assemblies, and controlling object visibility for ease of management of multiple layers, have evolved through such adoption of best practices.

We leveraged this phenomenon of evolving best practices by reviewing the current UI solutions that have emerged in 3D modeling design applications. Further, we used the best practices we observed in the course of the review to refine and characterize the UI design principles that were proposed in Section 3. In this section, we describe this process in detail.

4.1. A review process of 3D computer-aided design tools

The authors reviewed 10 state-of-the-art 3D computer-aided design (CAD) applications in an effort toward putting together a collection of UI-related best practices and developing principles to improve the UI issues identified from the survey described in Section 3 3D modeling systems were reviewed in the following steps:

(1) Choose reviewer: CAD systems are complex tools with many special functions that general computer users cannot operate. Thus, reviewers must be familiar with CAD systems as well as with general issues in human–computer interaction. The authors were the reviewers. The authors had over 10–30 years of experience with various CAD systems as a developer and user and also in the general field of human–computer interaction.

(2) Select systems to review: Since tools targeted for similar user groups tend to copy each other’s “good” features, the selection process should focus more on the number of types of tools than the number of applications to review.
We began with a list of software applications that lead the market and added systems known to have unique or novel functionality to the list, then short-listed the initial list through a series of discussions on the coverage of tool types and functionalities. All CAD tools chosen for the review were widely accepted in the US market, or still upcoming, but novel and holding promise. We reviewed the following systems: ArchiCAD, AutoCAD, Digital Project, DProfiler, Form Z, Maya, Revit, Rhinoceros, Solidworks, and VIZ. ArchiCAD and Revit are parametric modeling tools developed for architects. AutoCAD is a 2D and 3D drafting and modeling tool. Digital Project (an architectural version of CATIA) and Solidworks are parametric modeling tools initially designed for mechanical engineering. DProfiler is a quick modeling tool for early cost estimation. VIZ, Form Z and Rhinoceros are modeling and rendering tools. This list is not exhaustive and some tools with good reputation for their usability may have been omitted from this list.

Our goal was to extract principles from the best practices, not to enumerate every best practice, which are added to or replaced after each release cycle of software.

(3) Collect best-practice examples: Each reviewer chose three or four tools to review considering the availability and familiarity of software applications to reviewers. The review and selection process for these areas was qualitative and subjective. However, in order to gain internal consistency of the reviewed data between the reviewers, we tried to structure the review and selection process. General review guidelines and questions were provided. Reviewers were asked to review tools focusing on seven CAD UI problem areas identified from the case study described in Section 4.2: i.e., dialog box design, drawing generation, help, menu design, 3D modeling, toolbar layout, and viewing and navigation. For the review process, we designed a template to capture the UI behavior for each of the above broad areas. The review form with guide questions is provided in Appendix. For example, the template questions for 3D modeling were:

- How is creating new and opening existing models handled by this tool?
- What are the interesting 3D modeling features in this tool?
- Does the tool support creation of custom parametric objects? How does it handle specifying different parameters?
- How are properties of various components presented in the interface? The answers to these questions were recorded by the researchers for each application using screenshots, video clips and written explanations of the...
UI design. Additionally, the researchers also included their opinions about the quality, in terms of usability, of the design solutions they saw in the CAD tools (Fig. 3).

(4) **Review and generalize best practice examples:** Reviewers (the authors) went through multiple selection and review processes. Individual opinions regarding the collected examples that were jotted down by the researchers during the review were also considered collectively during the selection sessions. Ultimately, a set of best practices were selected from the 10 3D modeling applications in the form of a collection of possible solutions for the identified UI problems.

### 4.2. Best practices for UI problems

Here we list some of compiled best practices in each UI problem area. Some examples are proprietary features of one specific system, but other examples can be also found in multiple systems with slight variation in details.

#### 4.2.1. Dialog box design

Because all objects in 3D modeling systems are defined using a set of parameters, the interface for specifying these parameters is one of the most frequently performed tasks undertaken. Most commonly, it is done in a form-based environment composed with input widgets like textboxes, dropdown combo-boxes, radio buttons, etc., and action widgets like buttons. For the purpose of this study, we have called such interfaces *dialog boxes*.

In our review of best practices, we found that important dialog boxes (like the ones used to specify properties of objects) are integrated seamlessly into the design of the overall interface of the software. Unlike a floating interface found commonly in desktop applications, in 3D modeling tools, dialog boxes have a designated location on the screen—typically a vertical slot to the left or right of the screen, and are generally always visible or easily accessible at all times. If there are too many parameters that cannot fit on the screen at one time, they are categorized into tabbed groups designating sets of parameters associated with features. Alternatively, the dialog box can be made vertically scrollable. Fig. 4 shows a typical tabbed properties dialog box located in the vertical left slot of the screen, the width of which can be altered by the user. Although this interface decreases the size of the modeling window (the space to its right), it does not obstruct the view of the model.

We also found that dialog boxes display graphical icons and symbols to elucidate the meaning of parameters that may not be self-explanatory.

Some dialog boxes have a “preview window” that shows a simplified view of expected changes in the model when changed values are applied. This is a very important feature for complex 3D CAD systems for two reasons: (1) in case of a large model, some operations take a long time to calculate and regenerate the model; and (2) in a 3D system, some spatial options such as “top, bottom, left, and right” and combinations of them are very confusing to understand even with graphic icons and symbols because their definitions change depending on which side of a model users are seeing. For the same reasons, some systems show expected results of relatively “light-weight” operations (operations that require relatively simple calculation) directly in the model as a half-transparent view. As hardware becomes faster, this type of “direct preview” is expected to become common.

#### 4.2.2. Drawing generation

One of the important products of a 3D model in architecture, engineering, and construction (AEC) is a consistent set of drawings, to be used by architects, engineers, contractors, and the field fabricators and installers. Producing all drawings from a single integrated 3D model eliminates the chance of inconsistent views resulting from manual errors that exist in any large set of electronically drafted drawings. Not all 3D CAD applications we reviewed had automated drawing generation functionality. Hence, our observations in this category are based on only those systems that provided this feature. The
systems were ArchiCAD, Digital Project, Revit, and Solidworks.

In these types of systems, the relation between drawings and a 3D model is generally bidirectional in which any changes that are made to the model are automatically reflected in the drawings and vice-versa. This is possible because the drawings are specially formatted views of the model. All drawings are updated after every design action and are consistent with each other at any given time. All the systems that we reviewed supported the bidirectional operations between drawings and the model. Some systems in the market, however, do not allow users to make changes to the model from the drawings, to ensure that users do not make illogical or erroneous changes in 2D drawings without examining their impact in a 3D context. This is called unidirectional drawing update. The unidirectional drawing update has advantages and disadvantages. A disadvantage is that some changes require two levels of update: first to the model and then to the drawing annotation. Earlier generations of some systems did not maintain the relation between drawings and a 3D model and both unidirectional and bidirectional updates are big improvements.

There were two conceptual approaches to the creation of drawings—again both were effective and produced desired results. Some systems, provide predefined drawing views (e.g., “top” view or “plan” view) as the main starting point for modeling and drawing generating. As users add elements to the drawing view, a 3D model is created. Users can go back and forth between various drawing views and the 3D model view and complete the design. For example, building objects (e.g., walls, doors, windows, etc.), dimensions, symbols, tags, templates, etc. are added directly onto the “top” view of the model, and a 3D model will be completed. The view can be printed as a Plan drawing. Many systems originated from the AEC domain take this approach.

The other school of thought considers drawings as reports derived from the model. In these systems, the main starting point is any user-selected 3D view. Usually these systems have two distinct modes—a modeling mode and a drawing mode. Drawings are reports, like tables or spreadsheets that are extracted from the model. Drawing views are created by selecting model views like top view, north view, right view, etc. and are then “dropped into” the drawing mode “sheet” to create a drawing (like plan drawing, elevation drawing, sectional drawing, etc.). All properties related to drawings, such as drawing scale, dimension styles, and templates, are manipulated in drawing mode. The drawing-specific aspects are carried in the drawing across multiple sessions of “drawing mode” by associating them with the model entities that are linked to in the drawing.

4.2.3. Help

In most reviewed 3D CAD systems, we found that there were two kinds of help offered to the users. One is in the form of Help files that can be accessed when user explicitly requests for help. The other is more implicit and is provided automatically by the system in the context of the current user action.

**Help files:** In the survey many users pointed out problems with missing information or steps in Help files. Animated tutorials that some applications provided were very effective in filling in missing steps (Fig. 5).

The Help files that we judged as good were structured hierarchically and items at different levels or at the same level were cross-linked to each other. They generally had two main levels: the top level and the detail level. The top level included the general overview of the topic and
Table 2
A typical template used for organizing content on a detail-level page of Help files.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Displays the name of the function.</td>
</tr>
<tr>
<td>Icon</td>
<td>Displays a representative graphic that is also present in the toolbar.</td>
</tr>
<tr>
<td>Description</td>
<td>Explains what the function does and what results it produces.</td>
</tr>
<tr>
<td>Steps</td>
<td>Provides step-by-step instructions to execute it and are supported with many graphics.</td>
</tr>
<tr>
<td>Notes</td>
<td>Explains unique constraints and limitations of the described function.</td>
</tr>
<tr>
<td>Where on UI</td>
<td>Describes where this function can be found in the user interface. Typically accompanied by a screenshot of the interface.</td>
</tr>
<tr>
<td>Show me (opt.)</td>
<td>Provides hyperlinks to animated instruction files</td>
</tr>
<tr>
<td>Related topics</td>
<td>Provides hyperlinks to topics that are related to the current function.</td>
</tr>
</tbody>
</table>

provided hyperlinks to the relevant detailed sub-topics. The detail level displayed detailed descriptions on each function, chosen from among the hyperlinks in the top level of the topic. General contents of a detail level page include the items in Table 2.

The Help files of some applications had another way to structure content. Explanation for each concept was divided into two categories: Information and Tutorial. The Information section contained the glossary, explanation of what the tool is about and what is new in the tool. The Tutorial section provided instructions on how the tool is to be used. After searching for a specific command in the Help files, users could then choose between the two types of content layouts depending on their needs. Both approaches were considered effective.

**Context-sensitive help:** The context-sensitive help provides direct access to help or instructions to users. They could be categorized in three types:

- **Pop-up tooltips** displaying the label of an icon and a short description of how the command is to be used. The tooltip appears when the mouse pointer is held on it for a second or so.
- **Prompts for next steps** provided in a command line or at the bottom of the window to guide the user through multiple steps of an operation.
- **Prompts appearing close to the mouse** in an expanded cursor bitmap that displays a short description of the next step and provides dynamic input boxes where the user can type in parameter values while executing a command. (See Fig. 6.) All three context-sensitive help features were effective in aiding users to accomplish complex sequences of operations. Since the cursor is the main area of visual focus, cursor-related prompts require less eye travel, for example to the command line.

4.2.4. Menu design
The top-level menus in many applications that we reviewed were categorized by either distinct work phases in the project or by object type. We believe that such classification helps users to easily locate functions from within a possibly huge hierarchy of menus. For example, in one of the systems we reviewed, a typical project progresses in four phases: Laying out the site → Creating massing → Detailing the interior → Costing. The top-level menu structure of this application includes menus: Site, Massing, Interior and Costing.

In complex applications like most 3D parametric engineering tools, allowing users to customize various settings to their needs and preferences is generally helpful. We noticed that the design of menus of some of the reviewed systems addressed this issue in a couple of different ways. Some systems provided a dedicated menu, typically named “Customize”, which provided users with access to all the different customizable user settings for toolbars, hot keys, shortcuts, and other user preferences like background colors, etc. Others built the customizing ability for menus right into the UI of the menus. With this design it was possible to customize the contents of each menu by turning the items in it “on” or “off” using checkboxes. See Fig. 7.

As a rule, it is necessary to provide access to all (or most) functions available in an application through its menu structure. However, some parametric design tools offer such a large variety of functions that it becomes difficult to accommodate all of them in a single menu layout. In response, some systems implemented a multi-mode structure and displayed only menus specific to the currently selected mode. In such cases, it becomes necessary to provide obvious and easy ways for users to toggle between the different modes. Fig. 8 gives an example of one such interface.

Some systems we reviewed had unique interaction tools in addition to the conventional menus and toolbars to access the various commands. The purpose of these interaction widgets was to house the more commonly used features and to make them easily visible and accessible to the user. We found the following designs especially useful:

**Chest of drawers:** One of the 3D parametric modeling tools list commonly used menu items in an Option Bar on the far left of the screen that conceptually resembles a chest of drawers. The menu headings are listed vertically and
each menu can be slid “open” or “shut” by clicking on the heading to reveal its contents. Each item displays both, the name and the associated icon of the command, and is hence a hybrid between a toolbar icon and a menu item. Only one menu “drawer” can be opened at a time and opening a menu drawer automatically shuts all other drawers. This type of interface is useful because it puts commonly used commands within easy reach of the user, and takes up relatively less screen space than static toolbars.

**Shelf space**: The purpose of “shelf space” is similar to that of “chest of drawers”—to give easy access to commonly used tools, and occupy minimum space while doing it. The “shelf” is located on the top of the window, just below the menus and displays a group of command icons (like toolbars). However, different command groups can be dynamically selected to be displayed on the shelf space by choosing one of the many tabs available. In this design too, only one tab can be opened at a time. Refer to Fig. 9 for an example.

### 4.2.5. Toolbar layout

Most complex 3D parametric modeling systems provide a vast variety of tools to create, manipulate and analyze models. While menus across the top of the window play one critical role in applications, toolbars, configured in various ways, and often selected from a menu, carry most of the detailed operations. Often, there are so many tools offered that they quickly occupy the screen space and typically cannot be all displayed at the same time. While surveying the 10 systems, we tried to find solutions to address this problem of screen real estate. We found that this issue was tackled by the CAD systems broadly in one of three ways:

**Grouping**: The main toolbar, or command palette, consists of approximately 30 or so icons. Each icon in the palette is, however, a representative icon for a group of specific commands which are displayed when it is clicked. For example, a cube icon represents a group of commands that create primitive solids (like cube, sphere, cone, etc.). The set of commands will be displayed only when the user clicks on the cube icon. This design strategy is similar to the “chest of drawers” and “shelf space” designs discussed earlier.

**Collapsing toolbars**: Some applications allow users to “hide” or “show” entire toolbars by clicking on small collapse-expand icons (see Fig. 10). Although this is a good solution for clearing up a lot of screen space, we argue that the user might find it difficult to locate tools from collapsed toolbars, and hence might not be the best overall solution to the problem.

**Modes**: As in the case of menu design, multi-modal interfaces are sometimes used as a solution for accommodating large numbers of icons, especially when they can be neatly divided into independent logical categories. A few systems have created modes based on the level of experience of the user. Because many times expert users do not use toolbars at all, and instead prefer using keyboard commands, expert modes in some systems do not display toolbars at all.

### 4.2.6. 3D modeling

Because 3D modeling was the primary task of all the applications we were reviewing, we recorded a wide variety of best practice examples dealing with this area. Here we
elaborate some of the observations we made during the review process.

Creating and opening models: When it comes to creating new and opening existing models, we found that adhering to the conventions of the operating system and adopting procedures for creating and opening files in other applications on that OS, in general, reduces learning. However, we found that adding new functionality on top of the conventional design can be useful. For example, in one system we reviewed, when the “New” command is clicked while another model is already open, the system asks the user whether he/she would like to copy the objects in the open file into the new file. This is useful when the user might want to retain objects from the previous file and make only a few changes to them.

Selection: Objects are typically selected in 3D modeling environments using direct manipulation techniques like point and click. We found that selection was done either by clicking on one of the edges of an object when displayed in “wire-frame” mode, by clicking on one of the surfaces when displayed in “smooth rendering” mode, or by drawing and dragging a bounding box around one or many objects for multiple selection. The shape of the on-screen cursor changed when in the vicinity of an object, indicating that it is selectable.

We discovered that often color coding and highlighting was used to identify the selected objects from the ones not selected. Fig. 11 shows an example of color coding where selected vertices of a 3D mesh are colored red and the rest appear blue.

Automated checking: Some CAD applications have intelligent behavior built into them so that they can check the validity of certain commands while they are being executed. For example, in one system, when a closed two-dimensional shape (like a polygon) is created in a sketching tool, it is automatically filled with a solid color. This is a good check for situations where a user accidentally does not close the polygon properly—the shape simply does not turn solid color.

In other applications, some basic design rules and physical conflicts can also be automatically checked, where the system warns the user of potential problems and bad design.

Direct interaction with objects: Some systems allow users to directly sketch shapes on surfaces of existing solids in the model. Further, once drawn, these shapes can be extruded or subtracted from the solid by “grabbing and pushing” or “pulling” the surface (using a mouse pointer) to create new geometry. Refer to Fig. 12. Direct manipulation helps the user to easily modify model shapes.

Most 3D CAD tools also provide direct interaction with objects in the model for commands related to translation (movement), rotation and scaling. To enhance the functionality of these commands, some systems display an interactive icon or “gizmo”, which allows users to control axes and planes while performing the operations. Fig. 13 shows examples of gizmos.

Accessing commands from modeling space: Some applications allow users to dynamically access relevant commands based on current selections using mouse right-click or with certain combinations of keyboard buttons. Such context-sensitive access to commands without ever leaving the general modeling space allows an expert user to work at a considerably faster pace.

4.2.7. View and navigation

A view in 3D modeling applications can be imagined as a camera provided to the user to look at a model from a particular direction and to interact with it from that perspective. Views also can be adjusted with regards to displaying transparency, colors and other aspects of visibility. Most of the 3D modeling tools we reviewed provided some basic views such as “top view”, “bottom view”, “plan”, “elevation”, “3D view”, etc. as default views when a model is created. Further, they allowed users to create their own custom views if required. We found that absolute directional labels such as “north elevation”, “south view”, etc., based on a pre-defined north-point,
are best naming conventions for views, to avoid any confusion about orientation of the model.

We observed a variety of ways in which custom views were created. One application used interactive conventional drawing symbols or tags to represent custom made sections and levels. Adding a tag to the model essentially allowed the user to create a corresponding view at that location. Another system allowed the user to access predefined views from a drop down list. Icons with “+” and “−” signs located close to the list were provided to add or delete views from the list. A third system provided access to default views by clicking on appropriate icons (see Fig. 14).

Last, some 3D applications provided the concept of “layers” to allow users to group various objects, and manipulate them collectively. For example, an important function in all CAD applications is to control visibility of objects in a model, and this was accomplished by “hiding” or “showing” certain layers. Usually, manipulation of layers is controlled via a Layer Manager dialog box—an example is displayed in Fig. 15.

5. Findings and proposed UI principles

As an effort to improve the UI issues of a complex AE system, we reviewed 10 state-of-art CAD systems and compiled best practices of each UI problem area identified through a survey. We generalized the common factors from the collected best practices and proposed them as a set of UI principles for complex 3D design and engineering systems. The generalization process was heuristic.

We recognized that many criteria regarding user interaction are universally applicable to a number of different interactive desktop systems. Where applicable, we adopted names for some of the criteria from similar ideas found in the literature. However, we have modified their definitions to make them more specific and suitable to our software tool domain. For example, we borrowed the terms
Familiarity, Customizability, Consistency, and Recoverability, from Dix’s Principles to Support Usability (Dix et al., 1997) but reworded them so as to be more salient in our application domain. Broadly, we have categorized the principles into three areas based on the tasks that they support. The principles are cross-referenced with specific best practice cases discussed in Section 4.

5.1. Principles for general system design

This category deals with common UI principles also applicable to general system design. Thus, principles in this category overlap with existing UI principles for other types of systems.

Consistency: Uniformity of system semantics across similar situations. For example, using a consistent template for all dialog box designs through the whole system (e.g., the properties box in Fig. 4), consistent interaction methods and behavior for similar operations, or using terms in a consistent way with minimal use of synonyms or homonyms.

Visibility: Making relevant information conspicuous and easily detectable to the user. For example, making various options within operations readily visible (instead of requiring knowledge of function keys) (e.g., the properties box in Fig. 4 and context-sensitive help in Fig. 6); and tools easy to find, making sure that operations used together are easily accessed from each other and not hidden under several layers of hierarchy.

Feedback: Response of the system to the user’s actions in order to provide information regarding the internal state of the system. For example, highlighting a model object, or displaying a message that shows how many items are selected, immediately after the user applies the selection command, or indicating the status of a complex multi-step operation (e.g., color coding for selected elements in Fig. 11).

Recoverability: Providing the user with options to recognize and recover from errors. For example, providing undo and redo commands and automated checking described in the 3D modeling section.

5.2. Principles specific to 3D parametric design

Maximization of Workspace: Providing maximum screen space for carrying out the primary functions of the CAD system. Specifically, this addresses viewable space for modeling and viewing models and drawings (e.g., collapsing toolbars in Fig. 10). As the number of CAD users that use multiple monitors is increasing and also large monitors are getting cheaper, an effective way of separating workspace and menu space is an issue.

Graphical Richness: Replacing textual information with graphical information like imagery or animation to enhance user comprehension where appropriate. For example, providing preview for the effect of a certain function or thumbnail images of catalog files, color coding of selected objects, faces, edge, points, etc.

Direct Manipulation: Providing interaction that is perceived by the user as directly operating on an object or entity within the system. For example, allowing the function for modifying a shape by “dragging” a face of a solid object, “stretching” a line, or “pushing” a point on a surface (e.g., direct manipulation in Fig. 12).

5.3. Principles for user support

Familiarity: Leveraging user’s knowledge and experience in other real-world or computer-based domains when interacting with a new system: for example, conforming to existing system conventions like: operating system conventions, 3D modeling system conventions or conventions arising out of similarity with the users’ actual work environments, like drawing board metaphor or desktop metaphor (e.g., menu design based on typical design/engineering phases).

Customizability: Support to explicitly modify the interface or operability of the system based on the user’s preference. For example, allowing customization of toolbars and menus (e.g., Customize Menu in Fig. 7 and Custom Views in Fig. 15).

Assistance: Providing support to the user both explicitly, by tutoring, and implicitly, by prompting the user in the right direction. Example of explicit support—providing well-documented Help files and Tutorials. Example of implicit support—offering command line prompts and tool tips for all actions (e.g., Help files with animated instructions in Fig. 5 and context-sensitive help in Fig. 6).

Minimalist design: Keeping the design simple and minimizing redundancy of information when it threatens to be the cause of confusion to the user. For example, hiding or disabling controls in a dialog or menu, which are not legal or relevant for the current selection of objects in the model (e.g., a mode selector in Fig. 8).

Context recognition: Automatic adjustment of the interface or operability of the system based on user mode and, system context: for example, providing intelligent and context-sensitive Help to the user (Fig. 6).

6. A summary and discussion

Complex 3D design and engineering systems are usually composed of several hundred menu items. If options for each menu item are considered, the combination of possible operations grows exponentially. Since this number exceeds the cognitive load that a person can handle, an efficient and user-friendly UI is critical to the users of these systems.

We conducted a survey on the UI issues of a complex 3D design and engineering systems and collected 627 UI issue items through an on-line bug report system, questionnaires with trainees after the training sessions of the system, and focused group interviews with trainers. We summarized them as 179 distinct UI issues and categorized them into...
seven UI problem areas for 3D design and engineering systems including dialog box design, drawing generation, the Help structure, a command menu structure, 3D modeling, toolbar layout, and viewing and navigation. In order to find a way to improve the UI issues, we reviewed 10 state-of-the-art 3D CAD systems and collected the best practices in each UI problem area identified from the survey. The 10 systems are ArchiCAD, AutoCAD, Digital Project, DProfiler, Form Z, Maya, Revit, Rhinoceros, Solidworks®, and VIZ.

From the best practices reviewed, we generalized the common factors in the best practices and derived a set of user interface design principles reflecting the problems identified in complex 3D AE systems. The proposed UI principles are Consistency, Visibility, Feedback, Recoverability, Maximization of Workspace, Graphical Richness, Direct Manipulation, Familiarity, Customizability, Assistance, Minimalist Design, and Context Recognition. Some of them are unique to complex 3D systems. Some of them are generic and common UI principles that are applicable to other types of systems.

Some of these principles seem contradictory to each other. The principle for Maximization of Workspace suggests that maximum area should be allotted to primary functions such as creation and manipulation of the 3D models. Simultaneously, the Graphical Richness principle encourages the interface designer to include pictures and animations in interactive dialogs to better convey the meaning and purpose of the parameters displayed. This is generally not possible without occupying more space on the screen, and in turn either obstructing or reducing the main modeling area. In such circumstances, the UI designer has the dilemma of choosing strategies that balance the trade-off between seemingly contradictory design principles.

We suggest that the choice of the dominating criterion should depend on the task or set of tasks that the design is catering to. In the above example, if the task being designed is to set the basic (and self-explanatory) parameters of the selected model object (for example, height, width, material, etc.) the designer should choose the Maximization of Workspace as the dominating principle. On the other hand, if the parameters to be adjusted are complex and require explanatory graphics to convey their meaning, the designer should rely on the Graphical Richness principle as the main design principle. The final desired result will be a balancing of the competing criteria.

In this example, we indicate our view that the principles proposed here are meant to provide a framework to design or evaluate UIs for 3D parametric modeling applications. They are not hard and fast rules, but rather guidelines to help designers create consistently good designs, and may be altered or ignored if specific design scenarios demand it.

The elaborated UI principles are proposed as guidelines for improving and evaluating 3D parametric AE design tools in the future. Many of the user interface design principles we defined here also can be used in other areas, especially those involved in complex modeling tasks such as business process modeling or other graphical applications. In the end, these function-oriented principles must be adapted and integrated with more general ones dealing with issues supporting innovative thinking and creativity (Resnick et al., 2005; Shneiderman et al., 2005).

**Acknowledgments**

The authors are thankful to the user interface team at Tekla Inc., Helsinki, Finland, the Tekla Structures® training team at Tekla Inc., Kennesaw, GA, and the Tekla Structures® training team at Leap Software, Tampa, FL for their continuous support and feedback through this project.

**Appendix. Review form**

<table>
<thead>
<tr>
<th>Software:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
</tr>
<tr>
<td><strong>Main Issue</strong></td>
</tr>
<tr>
<td><strong>Menu Structure</strong></td>
</tr>
<tr>
<td>What is the general structure of the layout?</td>
</tr>
<tr>
<td>How does the menu structure differentiate between object creation and object modification?</td>
</tr>
<tr>
<td>How does the tool deal with a large number of commands?</td>
</tr>
<tr>
<td>Are there modes with different arrangement of menu items offered by the tool? What are they?</td>
</tr>
<tr>
<td><strong>View/Navigation</strong></td>
</tr>
<tr>
<td>Does the tool provide any default views? What are they?</td>
</tr>
<tr>
<td>Is there a way to add a custom view? How?</td>
</tr>
<tr>
<td>What are the hot-keys used by the tool? How to they compare to MS Windows conventional hot-keys?</td>
</tr>
<tr>
<td>What are the ways to toggle between views or select the right view?</td>
</tr>
<tr>
<td>How is the visibility of objects in views manipulated? (Use of layers, etc.)</td>
</tr>
</tbody>
</table>
Help

What is the general structure of the layout/template?
Does the tool support context-sensitive help? How?

Modeling

How is creating new and opening existing models handled by this tool?
What are the interesting modeling features in this tool?
Does the tool support creation of custom parametric objects? How does it handle specifying different parameters?
How are properties of various components presented in the interface?

Selection

How does the tool handle selection of various parts of the model (objects, lines, points, etc.)?

Toolbars

How is the problem of excessive number of icons in the interface handled by the tool?
Does the tool provide customizing facility for toolbars? How?
Are there different modes in the display of toolbars? What is the distinction?

Drawing

Does the tool support automatic creation of drawings? How are drawings created?
How are the properties of drawings edited (scale, dimensions, etc.)?
How are templates created and edited?

Others

References