

AUTOMATICALLY INCORPORATING THIRD PARTY FEATURES INTO A COMPUTER DESIGN SCHEMATIC

BACKGROUND OF THE INVENTION

5 The Field of the Invention

The present invention relates generally to computer-aided design or drafting software.

Background and Relevant Technology

As computerized systems have increased in popularity so have the range of applications that incorporate computational technology. Computational technology
10 now extends across a broad range of applications, including a wide range of productivity and entertainment software. Indeed, computational technology and related software can now be found in a wide range of generic applications that are suited for many environments, as well as fairly industry-specific software.

One such industry that has employed specific types of software and other
15 computational technology increasingly over the past few years is that related to building and/or architectural design. In particular, architects and interior designers (“or designers”) use a wide range of computer-aided design (CAD) software for designing the aesthetic as well as functional aspects of a given residential or commercial space. For example, a designer might use a CAD program to design
20 fixtures and furniture for a particular office. The designer might then export the designs to be manufactured by a particular millwork facility.

While millwork is becoming a more common method of producing furniture, producing custom millwork furniture can be an expensive and time-consuming process. For example, conventional systems may require that custom furniture first be

meticulously designed within a CAD program. Additionally, prior to designing the furniture or fixture within the CAD program, conventional systems may require that the specifications of the end product be exactly known ahead of time. For instance, a designer may need to know the exact dimensions of the object being designed, along
5 with the finishing features, such as joint type, hinge type, door sizes, etc.

Additionally, in at least some conventional systems, great expense is incurred if any design changes are made after the initial CAD model is created. If, for example, a designer discovers that an initial measurement was incorrect, the designer may need to adjust or even recreate the entire design manually, taking into account
10 the correct measurement. In addition, large cost can be incurred by simply switching from one millwork provider to another. For example, different millwork providers may use different joints, different hardware, different materials, materials of different dimensions, etc. As mentioned above, even slight changes such as these may require significant reworking the CAD design.

15 Accordingly, there are a number of problems in the art relating to modeling architectural elements within a CAD program and later manufacturing those elements with a millwork facility.

BRIEF SUMMARY OF THE INVENTION

Implementations of the present invention overcome one or more problems in
20 the art with systems, methods, and apparatus for automatically incorporating third party features into an architectural design. In particular, in at least one implementation of the present invention, an architectural design can automatically be configured to account for the specific features, specifications, and hardware of a particular millwork facility. The same architectural design may then be automatically

configured to incorporate the features, specifications, and hardware of a different millwork facility. Additionally, in at least one implementation, inconsistencies that are created by a particular specification, feature, or hardware can be automatically resolved.

5 For example, a method in accordance with at least one implementation for incorporating third party features into an architectural design can include generating an architectural design comprising one or more sub-components. The method can also include identifying an interface of a third party feature with at least one sub-component of the architectural design. Additionally, the method can include
10 receiving an indication of the third party feature. Further, the method can include automatically incorporating the third party feature into the interface. Further still, the method can include automatically resolving anomalies that the incorporation of the third party feature creates.

 In an additional or alternative implementation, a method can include
15 generating within a computer aided drafting program an architectural design for an architectural element. The method can also include identifying one or more interfaces for manufacturer specific components within the architectural element. Additionally, the method can include receiving a configuration list. The configuration list can comprise manufacturer specific components and manufacturer specific specifications
20 for a particular manufacture. Further, the method can include automatically identifying manufacturer specific components and manufacturer specific specifications from the configuration list that correspond with the one or more interfaces. Further still, the method can include automatically incorporating the manufacturer specific components into the architectural design.

These and other objects and features of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

5 In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof, which are illustrated in the appended drawings. It should be noted that the figures are not drawn to scale, and that elements of similar
10 structure or function are generally represented by like reference numerals for illustrative purposes throughout the figures. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

15 Figure 1 is a schematic illustration of a system for designing and manufacturing an architectural element in accordance with an implementation of the present invention;

 Figures 2A illustrates a spatial framework created in accordance with the inventive system illustrated in Figure 1

20 Figure 2B illustrates the spatial framework of Figure 2A, after a designer has inserted a cube splitter in accordance with an implementation of the present invention

 Figure 2C depicts the spatial framework of Figure 2B upon implementation of one or more facet splitters in accordance with an implementation of the present invention;

Figure 3 depicts a planar view of a spatial framework created in accordance with an implementation of the present invention;

Figure 4A depicts a spatial framework created in accordance with the present invention in which there are no joint anomalies;

5 Figure 4B depicts a similar spatial framework as that of Figure 4A, albeit with joint anomalies;

Figures 5A depicts an architectural design for a drawer in accordance with an implementation of the present invention;

10 Figure 5B depicts a front face of the drawer depicted in Figure 5A in accordance with an implementation of the present invention;

Figure 5C depicts a drawer bottom and side, albeit without a connecting joint in accordance with an implementation of the present invention;

Figure 5D depicts the drawer bottom and side of Figure 5C, now with a joint, in accordance with an implementation of the present invention;

15 Figure 5E depicts a storage unit and an oversized sink in accordance with an implementation of the present invention;

Figure 5F depicts a resized storage unit and an inset sink as depicted in Figure 5E in accordance with an implementation of the present invention;

20 Figure 6 depicts a finished rendering of an architectural element created in accordance with an implementation of the present invention;

Figure 7A illustrates a spatial framework created in accordance with the inventive system illustrated in Figure 1;

Figure 7B illustrates a second spatial framework embedded within the spatial framework of Figure 7A, created in accordance an implementation of the present invention;

Figure 8A depicts an oversized cabinet system and an alcove in accordance
5 with an implementation of the present invention;

Figure 8B depicts the cabinet system of Figure 8A automatically adjusted to fit within the alcove in accordance with an implementation of the present invention;

Figure 9A depicts an undersized cabinet system and an alcove in accordance with an implementation of the present invention;

10 Figure 9B depicts the cabinet system of Figure 9A automatically adjusted to fit within the alcove in accordance with an implementation of the present invention;

Figure 10 illustrates a flowchart of a series of acts in a method in accordance with an implementation of the present invention for incorporating third party features into an architectural design; and

15 Figure 11 illustrates another flowchart of a series of acts in a method in accordance with an implementation of the present invention for incorporating third party features into an architectural design.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Implementations of the present invention extend to systems, methods, and
20 apparatus for automatically incorporating third party features into an architectural design. In particular, in at least one implementation of the present invention, an architectural design can automatically be configured to account for the specific features, specifications, and hardware of a particular millwork facility. The same

architectural design may then be automatically configured to incorporate the features, specifications, and hardware of a different millwork facility.

For example, in at least one implementation of the present invention, a user can use an object oriented CAD program of the present invention to create a spatial
5 framework representative of an architectural element, for example, a desk with drawers. As the user is creating the framework of the desk with drawers, the CAD program can automatically create the surfaces and spaces that will make up the architectural element by analyzing the user's input. Additionally, the CAD program can identify potential uses for the spaces.

10 In at least one implementation, the space is identified and tracked by assigning an independently executable software object to the space. As an independently executable software object, the space can have independent functions and variables associated with it. As needed, these functions and variables can automatically adjust the space, and in turn the planes that define the space in response to manufacturing
15 specifications and manufacturer specific components.

Additionally, in at least one implementation, the present invention can automatically incorporate specific features and specifications from a third party manufacturer. In many cases, different manufacturers may use different hardware, different types of joints, and various other different manufacturing elements that each
20 may require unique adjustments to an architectural design. For example, when manufacturing a kitchen, a first millwork facility may use a smaller sink with only a single basin, while a second millwork facility may use a larger sink with two basins. In at least one implementation, the architectural design can automatically adjust to incorporate either the smaller sink or the larger sink. One will appreciate that the

ability to automatically incorporate third party features into an architectural design after the design is substantially complete can provide significant benefits.

Additionally, in at least one implementation, the present invention can aid in automatically resolving anomalies created by the incorporation of manufacturer
5 specific components and specifications into the architectural design. For example, a particular manufacturer may use a specific type of carpentry joint. Some combinations of joints, however, may result in an impossible configuration or a configuration with an anomaly – often at a corner. In at least one implementation, the present invention can automatically identify an impossible configuration or a
10 configuration that may create an anomaly and automatically resolve the conflict to create an architectural element with appropriate joints.

Implementations of the present invention can also allow a user to create a framework for an architectural element without knowing many of the end features that will be incorporated into the architectural element. For example, in at least one
15 implementation, a user can create a framework for an entire kitchen without knowing the material that will be used, the dimensions of the resulting kitchen, the specific fixtures, and many other similar details. Once the digital framework is completed a specific millwork facility can enter in the details of its materials, fixtures, configurations, and the actual specifications and/or dimensions of the end product,
20 and the digital framework can automatically adjust to conform to the entered information.

Figure 1 depicts an architectural schematic diagram of a computer system for designing and manufacturing an architectural element. In particular, Figure 1 shows a computer terminal 110 that is in communication with a millwork software application

100. The millwork software application 100 can be executed from the computer terminal 110, from a server (not shown) that the computer terminal 110 is accessing, or by using some other known method of execution.

The millwork software application 100 can comprise a plurality of modules
5 120, 130, 140, 142, 144, 146, 148, 160 that are adapted to aid in designing a file for millwork. In at least one implementation, the millwork software application 100 can comprise a user interface module 120, a manufacturing preparation module 130, a framework module 140, a facet module 142, a spaces module 144, a boundary module 146, a recursion module 148, a configuration list module 160, and a storage device
10 150. One will understand, however, that separating modules into discrete units is arbitrary and that modules that be combined, associated, or separated in ways other than shown in Figure 1 and still accomplish the purposes of this invention. Accordingly, the particular modules 120, 130, 140, 142, 144, 146, 148, 160 of Figure 1 are only shown for illustrative and exemplary purposes.

15 The user interface module 120 can be in communication with the computer terminal 110 through a series of data packets 112. For example, the user interface module 120 can display images and graphical controls to a user through a computer monitor and can receive input from a user through a keyboard and/or mouse. As a user creates and/or manipulates a particular framework of an architectural element,
20 the user interface module 120 can communicate to and receive instructions from the framework module 140. The framework module 140 can in turn communicate with the facet module 142, the spaces module 144, the boundary module 146, the recursion module 148, and the configuration list module 160.

Ultimately, either user interface module 120 or framework module 140 can communicate with manufacturing preparation module 130 to create a file that is prepared for use in a millwork facility. Additionally, the various modules can communicate with a storage device 150. The storage device 150 can contain, among
5 other things, templates for a variety of different designs, completed designs that can be used on a standalone basis or incorporated into other designs, tool lists and/or manufacturing information specific to particular millwork facilities, and/or particular design features.

One will appreciate in view of the specification and claims herein that the user
10 interface module 120 can provide to the user an option to create and make design changes to a framework 200. In at least one implementation, upon receiving a request for some modification, the user interface module 120 can communicate the request to the framework module 140. For example, a user may desire to design a desk for production at a millwork facility. Accordingly, a user may enter instructions into the
15 computer terminal 110 to design and create the desk. The user interface module 120 can in turn communicate those instructions to the framework module 140.

Upon receiving the instructions, the framework module 140 can communicate with the appropriate module to execute the request. For example, if the user desires to split the upper surface of the desk into two portions, the framework module 140 can
20 communicate with the facet module 142, which can modify and track surfaces within the framework 200. In contrast, if the user desires to split a space into two spaces, Figure 1 shows that the framework module 140 can alternatively use the spaces module 144, which can modify and track spaces within the framework 200.

For instance, the spaces module 144 can allow the user to split the framework 200 in half and create one half of the desk that is dedicated to drawers and another half that is open space for the user to place his or her chair and feet. Additionally, a user may use the spaces module 144 to split the framework 200 into any number of other divisions, for example thirds. In at least one implementation, the divisions do not need to be proportionally equal. For instance, the spaces module 144 can allow a user to move the single split mentioned above such that the drawers of the desk only take up one-third of the framework, while the leg space takes up the remaining two-thirds. In at least one implementation, the facet module 142 can perform similar functions on surfaces within the framework 200.

In addition, the framework module can use the boundary module 146 to automatically check joints within the framework 200 to determine if any “anomalies” exist. For example, if a user specifies that particular joint should be a miter joint, the boundary module 146 can analyze all of the joints within the desk to determine whether the remaining boundaries properly form around the entire desk. If the boundary module 146 detects any anomalies, (e.g., improperly overlapped joints, or other inappropriate positioning), the boundary module 146 can automatically resolve them and create proper joints throughout the desk.

Once the user has finished designing the architectural element (in this case a desk), the manufacturing preparation module 130 can receive millwork facility specific details from the configuration list module 160 and specifications relating to the final details of the architectural element. For example, a particular millwork facility may use a dovetail joint to assemble the drawers of the dresser. Additionally, the millwork facility may use a particular type of wood that comprises a specific

thickness. Upon receiving this information from the configuration list module 160, the manufacturing preparation module 130 can automatically adjust the framework 200 of the desk to create a design that incorporates the dovetail joint, wood type and wood thickness, and can be manufactured at the millwork facility. In contrast, in at least one implementation, the manufacturing preparation module 130 may not make any changes to the actual framework 200, but instead the manufacturing preparation module 130 may make the necessary changes to the actual manufacturing code (e.g., CNC code).

Similarly, the manufacturing preparation module 130 can automatically adjust the framework 200 of the desk to incorporate specific third party features. For example, a designer may originally design a desk without knowing the specific handles that a particular millwork facility uses for the drawers. In at least one implementation, the configuration list module 160 can receive a list of components and specifications that the particular millwork facility uses. The manufacturing preparation module 130 can then automatically incorporate those components and specifications of the millwork facility into the design.

For example, the millwork facility may use handles that require two pre-drilled screw holes spaced 10 cm apart with the midpoint between the holes being centered on the drawer face. Upon receiving this information from the configuration list module 160 the manufacturing prepared module 130 can automatically incorporate the two screw holes for the handle into the desk drawer. Additionally, the manufacturing preparation module 130 can automatically incorporate the holes into a computer numerical code (“CNC”) file that the particular millwork facility can use to automate the manufacture of the desk.

In at least one implementation, the manufacturing preparation module 130 can also automatically prepare the same framework 200 to be manufactured in any number of different millwork facilities, even though each facility may have specific manufacturing requirements. Once a particular framework 200 has been designed, the framework 200 can be used at a number of different millwork facilities as long as the configuration list module 160 has access to a configuration list that is associated with the specific facility.

Figures 2A-2C depict various implementations of a spatial framework 200. As shown in Figure 2A, the spatial framework 200 can initially comprise a simple cube. In at least one implementation, the millwork software 100 can comprise a plurality of simply shaped frameworks 200 that can be used as starting points for designing an architectural element. In general, the spatial framework 200 can be a computer model of an architectural element that captures the design intent of a user.

In particular, the spatial framework 200 can capture data relating to the outline of an architectural element and the position of components in the element with respect to each other. To accomplish this, the spatial framework 200 can comprise a space 250 that is associated with an independently executable software object. The independently executable software object can assist in tracking and managing the various components of the designed architectural element. In at least one implementation, the space 250 and independently executable software object are both managed by the spaces module 144.

By way of explanation, an independently executable software object comprises a set of computer-executable instructions used in object-oriented program code, and which relate to a particular physical component or feature. In addition,

software objects can be interrelated via parent/child dependency relationships where changes in a parent object flow through to a child object and vice versa. For example, a software object created for a table may have several child objects for each leg.

In other cases, the software objects can be related to other software objects that represent physically proximate components (e.g., a wall object that is positioned next to the table object). For example the above-mentioned table software object and leg software objects can independently execute in a correlated fashion to ensure each corresponding physical component (i.e., the table top, or the table legs) is positioned appropriately, or otherwise colored and designed consistent with the user's specifications.

Figure 2A also depicts that the framework 200 can comprise "facets" 202, 204, 206 and boundaries 208. By way of explanation, "facets" 202, 204, 206 represent surfaces within the framework. Facets 202, 204, 206, however, may not always correlate to surfaces within the finished architectural element. For example, facets 202, 204, 206 may only be quasi-two-dimensional because they can comprise a specified thickness. In some cases, a user can set the thickness of a particular facet to be zero. As a result the facet can still be a part of the framework 200 but it will not be a part of the finished architectural element. "Boundaries," on the other hand, represent lines where facets meet. In at least one implementation, a specific boundary's location can be defined with respect to the other boundaries that the specific boundary intersects.

Figure 2B depicts an implementation of a framework 200 that has been bisected by a "cube splitter" 210. As depicted, the cube splitter 210 splits or divides space 250 in half creating two new spaces 212 and 214. In at least one

implementation, the creation of two new spaces 212, 214 also results in the creation of two new independently executable software objects associated with each space 212, 214. It should be understood that while cubes and squares are used to illustrate embodiments of the present invention within this application, in at least one
5 implementation, many different shapes and configurations of a framework 200 can be used.

In particular, in at least one implementation, the newly created independently executable software objects associated with spaces 212 and 214 may each inherit the parameters and characteristics of the independently executable software object that
10 was originally associated with space 250. In at least one implementation, due to this inheritance, if space 250 originally comprises a set of drawers, after the split, spaces 212 and 214 can each automatically comprise a set of drawers that mirror the original drawers of space 250.

The cube splitter 210 can also create a new facet 216 within the framework
15 200 and a plurality of new facets (for example 220, 222) on each external surface of the framework 200. As mentioned above, the new facets 216, 220, 222 can each comprise a unique thicknesses such that the facets 216, 220, 222 comprises physical surfaces within the architectural element, or the facets 216, 220, 222 can comprise thicknesses of zero, resulting in the facets 216, 220, 222 only being represented within
20 the framework 200 but not within the finished architectural element.

Figure 2C depicts the framework of Figure 2B comprising two facet splitters 230, 232. As depicted, the facet splitters 230, 232 can split facet 222 into three new facets 234, 236, 238. In at least one implementation, a facet splitter 230, 232 can split only facets 220, 222, 234, 236, 238, as opposed to a cube splitter 210, which can split

an entire space 250, 212, 214. In addition, in at least one implementation, the new facets 234, 236, 238 remain associated with space 214 and thus can be associated with the independently executable software object that is associated with space 214.

Figure 3 depicts a quasi-two-dimensional view of front face 204 of the framework 200 from Figure 2C. As mentioned previously, the view is quasi-two-dimensional because each facet can in fact comprise a thickness. Figure 3 shows facet 220, which was formed by the placement of cube splitter 210, and facets 234, 236, and 238, which were formed by the placement of facet splitters 230 and 232. Additionally, Figure 3 depicts the end point 300, 301, 302, 303, 304, 305, 306, 307, 320, 322 of each boundary within the front face of the framework. In at least one implementation, a user can interact with the framework 200 through either a three-dimensional view (e.g., Figs. 2A-2C) or through a quasi-two-dimensional view (e.g., Fig. 3). In either view, the location and behavior of the facets can be managed by the facet module 142.

In at least one implementation, the location and positioning of the cube splitters 210 and facet splitters 230, 236, 238 within the framework can be tracked with respect to the end points 300, 301, 302, 303, 304, 305, 306, 307, 320, 322 of each respective facet splitter 230, 236, 238 or cube splitter 210, and in particular, where those end points intersect other boundaries. In at least one implementation, the location of the end point intersections can be tracked as a finite distance or as a proportion of the total length of the respective boundary.

For example, facet splitter 230 comprises end points 301 and 305. In at least one implementation, the location of facet splitter 230 can be designated as end point 301 being located distance 310 from the top of cube splitter 210 and end point 305

being located distance 312 from the top of side boundary 340. Similarly, the location of facet splitter 232 can be designated as end point 302 being positioned $\frac{1}{3}$ up the length of cube splitter 210, and similarly, end point 306 being location $\frac{1}{3}$ up the length of side boundary 340.

5 One will understand how similar measurement schemes can be used to locate and position any number of facet splitters and/or cube splitters within a framework 200. Additionally, one will understand that using a finite length or a proportional length can impact the future millwork of the architectural element. For example, if a user resizes an item framework 200 by expanding the framework in all directions,
10 then facet splitter 230 can still be located distance 310 and 312 from the top of cube splitter 210, and from the side boundary 340 respectively. Facet splitter 232, on the other hand, can change in absolute position such that each end point 302, 306 is $\frac{1}{3}$ up the length of their respective boundaries 210, 340.

 In at least one implementation, the framework 200 can be shrunk so much that
15 absolute distance 310 and distance 312 exceed $\frac{2}{3}$ of the total length of boundaries 210 and 340 respectively. One will understand that this can cause facet splitter 230 to overlap facet splitter 282. In this situation, the framework module 140 can automatically determine that either facet splitter 232 or facet splitter 230 should automatically be removed leaving only a single facet splitter 232, 230. For example,
20 in at least one implementation, a user can set an option to automatically give fixed lengths 310, 312 priority over proportional lengths 303, 316 or to automatically give proportional lengths 303, 316 priority over fixed lengths 310, 312.

 Additionally, a user may be able to set an option that gives priority to the first facet splitter 230, 232 or cube splitter 210 created over subsequent facet splitters 230,

232 or cube splitters 210. In contrast, a user may be able to set an option that gives priority to the last facet splitter 230, 232 or cube splitter 210 created over previous facet splitters 230, 232 or cube splitters 210. Further, in at least one implementation, a user can specifically designate that a particular facet splitter 230, 232 or cube splitter 210 should be given priority over other facet splitters 230, 232 and/or cube splitters 210.

Allowing a user to determine whether a facet splitter or cube splitter should be located on a proportional distance basis or on an absolute distance basis can provide the user with significant control over how an architectural element can be resized and manipulated. Additionally, allowing a user to determine the priority that particular cube splitters and/or facet splitters can be given when the splitters conflict with each other allows a user to have control over the final configuration of an architectural element that has been resized.

Figures 4A and 4B depict various implementations of boundary detection and correction. In at least one implementation, as a user designs a framework 200 for a particular architectural element, the user is able to specify the type of joints that the user desires to join particular surfaces of the architectural element. Similarly, in at least one implementation a particular millwork facility may use a specific joint type in some architectural elements. A user may be able to adjust or specify select joints, while others are fixed. As the user specifies the specific joints and the location of the specific joints, the boundary module 146 can analyze the framework to verify that no anomalies exist within the designated joints.

For example, Figure 4A depicts a simple architectural element that comprises a visible upper surface 400, a visible side surface 404, and a visible front surface 402.

As depicted, the front surface 402 and the side surface 404 meet each other at miter joint 420, the side surface 404 and the upper surface 400 meet each other at joint 430 where upper surface 400 overlaps side surface 404, and upper surface 400 and front surface 402 meet each other at joint 410 where upper surface 400 overlaps front surface 402. In at least one implementation, the boundary module 146 can analyze the joints of Figure 4A and determine that the above-recited joints do not create any anomalies, and can thus be left as they are.

In contrast to Figure 4A, the initial joints of Figure 4B do create an anomaly 440. In particular, Figure 4B shows that top surface 400 meets front surface 402 at joint 412 where front panel 402 overlaps upper surface 400. Additionally, Figure 4B shows that front surface 402 meets side surface 404 at joint 422 where side surface 404 overlaps front surface 402. Further, Figure 4B shows that side surface 404 meets upper surface at joint 432 where upper surface 400 overlaps side surface 404.

In at least one implementation, the boundary module 146 can analyze the joints of Figure 4B and identify the presence of anomaly 440. Additionally, in response to identifying the presence of anomaly 440, the boundary module 146 can resolve the anomaly 440 by automatically adjusting the joints. For example, the boundary module 146 can change boundary 422 such that the front panel 402 overlaps the side panel 404. One will appreciate that this change in the joint configuration will resolve the anomaly.

In determining what boundaries to change, the boundary module 146 can operate such that the joints specified by a configuration list from a millwork facility are preserved while other joints are changed. In contrast, the boundary module 146 can adjust the joints such that the most recently specified joint is preserved and others

are changed. Alternatively, the boundary module 146 can change joints such that the earliest specified joints are preserved and the most recently specified joints are changed. In addition, in at least one implementation, a user can specify that a particular joint be given priority over other joints.

5 In at least one implementation, the boundary module 146 can allow a millwork facility to specify particular types of joints for constructing an architectural element. A designer, therefore, can design the complete architectural design without being aware of the particular specifications of the millwork facility that will manufacture an architectural element. Additionally, a designer can create a single architectural design
10 that the manufacturing preparation module 130 can automatically adjust, as dictated by a millwork facility specific configuration list provided to the configuration list module 160. In this way a single architectural design can be used by a variety of different millwork facilities, each of which requires unique joints and specifications.

 Once an architectural element has been sufficiently designed within a
15 framework 200, the manufacturing preparation module 130 can prepare the framework 200 for actual production at a millwork facility. For example, the configuration list module 160 can provide the manufacturing preparation module 130 with the various specifications, hardware components, and other manufacturing constraints of a particular millwork facility. In at least one implementation, this can
20 include adjusting the framework 200 to incorporate a specific material type, or a specific material thickness, adjusting the framework 200 to fit within a particular space, incorporating the appropriate third party hardware into the design, incorporating the appropriate attachments into the framework, or adjusting some other portion of the framework 200.

For example, Figure 5A depicts a portion of a framework 200 representing a drawer 500. Specifically, the depicted drawer 500 comprises a backside 522, a left side 532, a right side 530, a front side 520, and a bottom 510. In at least one implementation, the manufacturing preparation module 130 can identify that a front side 520 comprises a third party hardware interface area 540 (i.e., a location where a handle can be attached). In response, and to identify the third party hardware interface area 540, the manufacturing preparation module 130 can query the configuration list module 160 to identify specifications for a handle that is used by a millwork facility of interest.

10 In at least one implementation, the configuration list module 160 can access a plurality of different configuration files from different millwork facilities stored on a storage device 150. A designer or user can indicate to the millwork software 100 the particular millwork facility that the user intends to use. The configuration list module 160 can then access the configuration list that is associated with the identified millwork facility. The various configuration lists can be stored within the storage device 150.

Once the configuration list module 160 provides the manufacturing preparation module 130 with the appropriate information, the manufacturing preparation module 130 can identify that the desired millwork facility utilizes handles that are anchored by two screws a certain distance apart. Based upon this information the manufacturing preparation module 130 can place two holes that are the appropriate distance apart within the third party hardware interface area 540 of the front side 520 as shown in Figure 5B. In at least one implementation, the

20

manufacturing preparation module 130 automatically incorporates the changes into a CNC file, or some other format that the desired millwork facility can interpret.

In addition to incorporating millwork facility specific components into an architectural design, in at least one implementation the manufacturing preparation
5 module can incorporate millwork facility specific joints into an architectural design. For example, the configuration list module 160 may identify that a particular millwork facility uses a groove joint to attach the bottom 510 of a drawer 500 to the right side 530 of the drawer 500. Upon receiving this information from the configuration list module 160, the manufacturing preparation module 130 can
10 automatically incorporate the groove joint 532 into the right side 530 of the drawer 500.

Figure 5D depicts the drawer of Figure 5C after the manufacturing preparation module 130 has applied the specified connection type 532 to the drawer. For example, the connection 532 can comprise the bottom side 510 being inset into the
15 right side 530. Upon determining and applying the proper connection type, the manufacturing preparation module 130 can create a file that will direct a millwork facility to cut a groove into the side wall of right side 530 at such a depth and location that the joint functions as designed in Figure 5D.

In at least one implementation, the configuration list provided by the
20 configuration list module 160 can include, among other things, the particular specifications and types of hardware that a particular millwork facility uses, the type of materials and thicknesses of materials that a millwork facility uses, and the types of joints that the millwork facility uses. Additionally, the configuration list can include information associating at least a portion of the items within the list with particular

elements of architectural designs. For example, the configuration list can contain an entry that associates a particular handle with any interface 540 on a drawer front panel 520. Similarly, the configuration list can comprise an entry that associates a groove joint with any interface where a right side drawer panel 530 meets a bottom drawer panel 510.

Additionally, in at least one implementation, a configuration file can provide multiple options for, among other things, hardware components, materials type and thickness, and joints. For example, a particular millwork facility may have a plurality of different drawer handles available for use. The configuration file can contain specifications and information about each available handle. The manufacturing preparation module 130 can then provide a user with various options to determine which components and specifications should be incorporated into a corresponding CNC file.

Figure 5E depicts a storage unit 600 that has been designed with an interface 552 for a sink 550. The depicted storage unit 600 comprises a cupboard 602 and three drawers 604, 606, 608. In at least one implementation, a designer can design the storage unit 600 without knowing the specifications for the sink 550 that will eventually be included with the storage unit 600. Additionally, the designer can design the storage unit 600 with the intent that the storage unit 600 be manufactured by a plurality of different millwork facilities, with each millwork facility using a unique sink 550.

The configuration list module 160 can provide the manufacturing preparation module 130 with the appropriate specifications for the depicted sink 550. As depicted, the sink 550 is too large for the specified interface 552. In at least one

implementation, the manufacturing preparation module 130 can resolve this discrepancy by automatically adjusting the size and configuration of the storage unit 600. For example, Figure 5F depicts an adjusted storage unit 600 that now includes the sink 550.

5 In the illustrated case, the manufacturing preparation module 130 lengthened the cupboard 602 of the storage unit 600 such that it now includes two cupboard doors 602a and 602b. The manufacturing preparation module 130 also narrowed the drawers 604, 606, 608 to compensate for the enlarged cupboard 602. Due to the changes that were automatically made to the storage unit 600 by the manufacturing
10 preparation module 130, the sink 550 now fits. In addition to reconfiguring the storage unit 600, the manufacturing preparation module 130 can also enlarge the interface 552 such that the sink 550 fits. The manufacturing module can also make the changes to a CNC file, or equivalent, such that the specified millwork facility can automatically manufacture the resolved storage unit 600 and sink 550.

15 In at least one implementation, when resolving inconsistencies within a particular architectural design the manufacturing preparation module 130 can rely upon a series of predefined constraints. For example, the manufacturing preparation module 130 can be directed to leave unchanged the external boundaries of a particular architectural element. For instance, the storage unit 600 and sink 550 from Figure 5F
20 can have the same external specifications as the original storage unit 600 from Figure 5E. One will understand that in some cases adjusting the external boundaries of the storage unit 600 can prevent the storage unit 600 from fitting in the location for which it was designed.

Additionally, in at least one implementation, the manufacturing preparation module 130 can be directed to adjust an architectural design to incorporate the hardware components specified by the configuration list module 160. For example, one will understand that in general a sink 550 cannot be placed directly over a drawer 604 because the sink will extend too deeply into the storage unit 600. Accordingly, in at least one implementation, the manufacturing preparation module 130 can identify that the cupboard 602 (and not the drawers 604, 606, 608) should be expanded to allow the sink 550 to fit.

In determining how to resolve inconsistency within an architectural design, in at least one implementation the manufacturing preparation module 130 can receive direction from at least one independently executable software object associated with a space 212, 214 within the framework. For example, as described above, a particular space 212 may identify itself as a cupboard 602, while another space 214 may identify itself as drawers 603, 606, 608. Additionally, the independently executable software object associated with the drawers can indicate that an object, such as a sink 550, should not be placed in the same space 214 as the drawers. In contrast, the independently executable software object associated with the cupboard 602 can indicate that an object such as a sink 550 can be placed in the same space 212 as the cupboards.

Once a user has finished designing an architectural element the manufacturing preparation module 130 can generate one or more CNC codes (or equivalent) that corresponds with or otherwise describes the architectural element. In an alternate implementation, the manufacturing preparation module 130 can generate parameters that are exported to a post processor that generates an appropriate CNC code. Once

the appropriate CNC code is generated a millwork facility can use the code to create the designed architectural element.

Figure 6 depicts a finished rendering of the architectural element 200. In particular, Figure 6 shows that the storage unit 600 comprises a cupboard 602 that correlates with the space 212 that was create by the placement of cube splitter 210. In addition, Figure 6 shows that the cupboard 602 comprises a door 610, which, for example, can be associated with facet 220. In addition, the storage unit 600 comprises three drawers 604, 606, 608 that were created by the combined placement of the cube splitter 210 and the facet splitters 230 and 232. As depicted, facet 234 can be associated with drawer 604, facet 236 with drawer 606, and facet 238 with drawer 608. In at least one implementation, additional designing that was not depicted directly by this application may have also been added to the framework 200. For example, the drawer 500 of Figures 5A-5D may have also been designed and added to the framework 200 of Figure 6.

Additionally, in at least one implementation, the system can automatically add gaps to the various components of a framework 200 such that features like drawers 234, 236, 238 and doors 610 are easy to open and close and are not overly snug. In some cases, the gaps may comprise slight millimeter spaces that are incorporated around the edges of a particular facet 212, 234, 236, 238. One will understand that if certain components of architectural elements are not designed and built with a gap, the component may not function or may function poorly, and such gaps can be easy to overlook during a conventional design face.

In addition to the foregoing, in at least one implementation, the storage device 150 and/or a configuration file provided by the configuration file module 160 can

contain visual information relating to various third party hardware that specific millwork facilities use. As such, in at least one implementation, the user interface module 120 can render a depiction of the architectural element, in this case the storage unit 600, displaying the unit as it will appear in its final form, including the correct connection types and third party hardware.

In at least one implementation, once a user has design an architectural element, the user can store the design within the storage device 150 for later access. For example, in at least one implementation, this allows the user to incorporate the architectural element into a new design. In particular, the designed framework can be recursively linked to an independently executable software object within another framework.

Figures 7A and 7B depict implementations for incorporating one framework 200 into another. For example, Figure 7A depicts a master framework 700 that includes a counter 710, three upper spaces 702, 704, 706, and two larger lower spaces 730, 732. In at least one implementation, the framework module 140 can associate a distinct framework 200 with a space 730, 732. Specifically, the space module 144 can use the recursion module 148 to associate a distinct framework with the independently executable software object that is associated with the space 730, 732. In other words, in at least one implementation, the system associates each space 730, 732 within a framework 700 with an independently-executable software object, which can recursively reference a copy of another distinct framework 200. As previously mentioned, an independently executable software object comprises a set of computer-executable instructions used in object-oriented program code, and which relate to a particular physical component or feature.

Returning to Figures 7A and 7B, a user can associate the two distinct frameworks 200 each representing the storage unit 600 from Figure 6 with spaces 730 and 732 respectively. Figure 7B depicts the resulting master framework 700 that includes spaces 702, 704, and 706, which can be designed into cupboards, and spaces
5 730 and 732, which both now contain frameworks 200 that are associated with storage units 600.

In at least one implementation, associating distinct frameworks 200 with spaces 730, 732 within a master framework provides a user with tremendous power and flexibility in creating a design. For example, each framework 200 can
10 independently access a framework module 140 and all other associated modules 142, 144, 146, 148. This can allow a framework 200 to dynamically and automatically adjust to any changes that are made to a master framework 700.

Additionally, in at least one implementation, a storage device 150 can comprise a framework library of pre-designed architectural elements. Each of these
15 stored frameworks can be associated with one or more independently executable software objects that can be recursively linked to other frameworks. For example, a designer can design an office space by accessing a group of stored frameworks that represent shelving units, desks, filing cabinets, cupboards, drawers, etc.

Once a designer identifies particular stored frameworks that the designer
20 wants to use, the designer can simply insert the chosen framework into a space within a master framework. Using pre-designed frameworks a designer can create a master framework that represents an entire office. Additionally, because the entire office was designed using spatial frameworks associated with independently executable software objects, the entire office design can change automatically to account for different

sizes, materials, features, etc. and such changes will correctly propagate throughout the design.

For example, Figure 8A depicts the cabinet system 800 designed in Figures 7A and 7B. The cabinet system 800 comprises a length 810, and a receiving wall 830
5 comprises a length 820. As depicted, length 810 is significantly longer than length 820 of the receiving wall. In at least one implementation, a user can specify that the length of the cabinet system 800 should be length 820, and the framework module 140 can automatically adjust the length of the cabinet system 800 to be length 820, while at the same time automatically and correctly adjusting all of the features of the
10 cabinet system.

Figure 8B depicts an implementation of an adjusted cabinet system 800. In this implementation, the framework module 140 automatically removed a cupboard and one of the storage units. In at least one implementation, the removal of the cupboard and storage unit may be a result of using absolute measurements when
15 tracking the location and behavior of the cube splitters 210 and facet splitters 230, 232 as described with respect to Figure 3. In particular, the framework module 140 may have identified that the storage unit length was reduced so much that there was no longer room to place the cube splitters 210 and facet splitters 230, 232 as was initially specified. Accordingly, the framework module 140 can automatically determine that
20 because the original length specifications cannot be met, a cupboard and a storage unit should be removed.

As an alternate example, Figures 9A and 9B depict an architectural element being expanded to fit a particular specification. For example, the cabinet system 800 comprises a length 810 and a receiving wall 930 comprises a length 910. As depicted,

length 810 is significantly smaller than length 910 of the receiving wall 930. As stated above, in at least one implementation, a user can specify that the length of the cabinet system 800 should be 910, and the framework module 140 can automatically adjust the length of the cabinet system 800 to be length 910 while at the same time
5 automatically and correctly adjusting all of the features of the cabinet system.

Figure 9B depicts an implementation of an adjusted cabinet system 800. In this implementation, the framework module 140 automatically expanded the length of the storage units 600 and added double doors 940 to each of the three cabinets. In at least one implementation, the expansion of the storage unit may be a result of using
10 proportional measurements when tracking the location and behavior of the cube splitters 210 and facet splitters 230, 232 as described with respect to Figure 3. In particular, the framework module 140 may have identified that the cube splitter between spaces 730 and 732 was specified as being placed at half the length of the bottom most boundary. As such, the framework module 140 simply expanded the
15 storage units to fill the larger length.

With respect to the cupboards, in at least one implementation, the framework module 140 and/or the manufacturing preparation module 130 can automatically identify some finishing features, such as, for example, whether a cupboard is narrow enough to only require a single door, or so wide as to require double doors 940. In the
20 depicted example, the framework module 140 and/or the manufacturing preparation module 130 determined that double door were appropriate due to the increased length of the cabinet system 800.

Accordingly, Figures 1-9 and the corresponding text illustrate or otherwise describe one or more components, modules, and/or mechanisms for incorporating

third party features into an architectural design. One will appreciate that implementations of the present invention can also be described in terms of methods comprising one or more acts for accomplishing a particular result. For example, Figures 10 and 11 and the corresponding text illustrate flowcharts of a sequence of
5 acts in a method for incorporating third party features into an architectural design. The acts of Figures 10 and 11 are described below with reference to the components and modules illustrated in Figures 1-9.

For example, Figure 10 illustrates that a method for incorporating third party features into an architectural design can comprise an act 1010 of generating an
10 architectural design. Act 1010 includes generating an architectural design comprising one or more sub-components. For example, Figures 2A-2C show various implementations of a millwork software application 100 being used to create an architectural design that comprises one or more sub-components.

Figure 10 also shows that the method can comprise an act 1020 of identifying
15 an interface. Act 1020 includes identifying an interface of a third party feature with at least one sub-component of the architectural design. For example, Figure 5A show that an identified interface 540 for a drawer handle.

Additionally, Figure 10 shows that the method can comprise an element 1030 of identifying a third party feature. Element 1030 can include receiving an indication
20 of the third party feature. For example, Figure 1 and Figures 5A-5F, along with the accompanying description, describe receiving an indication of third party features (e.g., drawer handle and sink).

Furthermore, Figure 10 shows that the method can comprise an element 1040 of incorporating third party features. Element 1040 can include automatically

incorporating the third party feature into the interface. For example, Figures 5E and 5F depict a sink automatically being incorporated into an interface 552.

Further still, Figure 10 shows that the method can comprise an element 1050 of resolving anomalies. Element 1050 can include automatically resolving anomalies
5 that the incorporation of the third party feature creates. For example, Figures 5E and 5F depict a stored unit 600 being automatically adjusted to fit a particular sink 550.

Additionally, Figure 11 shows that a method incorporating third party features into an architectural design can comprise an act 1110 of generating an architectural design. Act 1110 includes generating within a computer aided drafting program an
10 architectural design for an architectural element. For example, Figures 2A-2C show various implementations of a millwork software application 100 being used to create an architectural design.

Figure 11 also shows that the method can comprise an act 1120 of identifying one or more interfaces. Act 1120 includes identifying one or more interfaces for
15 manufacturer specific components within the architectural element. For example, Figure 5A depicts an identified interface 530 for a drawer handle.

Additionally, Figure 11 shows that the method can comprise an act 1130 of receiving a configuration list. Act 1130 includes receiving a configuration list, wherein the configuration list comprises manufacturer specific components and
20 manufacturer specific specifications for a particular manufacturer. For example, Figure 1, Figures 5E and 5F, along with the accompanying description, describe and depict the use of a list that is specific to a particular manufacturer.

Furthermore, Figure 11 shows that the method can comprise an act 1140 of identifying manufacturer specific components. Act 1140 includes automatically

identifying manufacturer specific components and manufacturer specific specifications from the configuration list that correspond with the one or more interfaces. For example, Figures 5A-5F depict various implementations of the present invention identifying interfaces for various components (e.g., drawer handle, drawer joint 532, and sink 550).

Further still, Figure 11 shows that the method can comprise an act 1150 of incorporating manufacturer specific components. Act 1150 includes automatically incorporating the manufacturer specific components into the architectural design. For example, Figures 5A-5F depict various implementations of the present invention automatically incorporating various components into the architectural designs (e.g., drawer handle, drawer joint 532, and sink 550).

Accordingly, Figures 1-11 provide a number of components, schematics, and mechanisms for incorporating third party features into an architectural design. Additionally, one or more implementations can allow a designer to develop an architectural element, such as a desk or even an entire kitchen, without knowing the specific specification and components that are used by a particular millwork facility. For example, a user can design a kitchen, including cabinets, drawers, counter tops, sink locations, etc., without knowing the materials that will be used to construct the cabinet and counter, or the final third party elements, such as the actual sink type, knob and handle configurations, wood joints, etc. One will appreciate that implementations of the present invention provide tremendous flexibility and power to designers and millwork facilities by allowing designers to construct detailed and specific schematics that can automatically be adjusted to configured with any number of millwork facilities.

The embodiments of the present invention may comprise a special purpose or general-purpose computer including various computer hardware components, as discussed in greater detail below. Embodiments within the scope of the present invention also include computer-readable media for carrying or having computer-executable instructions or data structures stored thereon. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer.

By way of example, and not limitation, such computer-readable media can comprise RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to carry or store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a computer-readable medium. Thus, any such connection is properly termed a computer-readable medium. Combinations of the above should also be included within the scope of computer-readable media.

Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or

acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are
5 to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

1. In a computerized environment, a computerized method for incorporating third party features into an architectural design, the method comprising:

generating an architectural design comprising one or more sub-components, wherein each of the one or more sub-components is associated with a different independently executable software object that is unique to each sub-component selected from the one or more sub-components and each different independently executable software object comprises independent variables and functions that are specific to different architectural elements that are represented by each sub-component selected from the one or more sub-components;

identifying a hardware interface of a third party feature with at least one sub-component of the architectural design, wherein the hardware interface is configured to receive the third party feature;

receiving an indication of specifications of the third party feature; and

automatically incorporating the specifications of the third party feature into the hardware interface, wherein:

the hardware interface is adaptable to receive a plurality of different third party features,

a particular independently executable software object that is unique to the at least one sub-component of the architectural design provides direction about the correct placement of the third party feature with respect to the hardware interface, wherein the particular independently executable software object inherited its characteristics from a parent independently executable software object from which the particular independently executable software object was split;

the specifications of the third party feature are the particular specifications of a millwork facility that will manufacture the hardware interface, and

automatically incorporating the specifications of the third party feature comprises automatically configuring a numerical control file to incorporate the specifications of the third party feature into the hardware interface.

2. The method as recited in claim 1, wherein identifying the interface comprises:

identifying a first sub-component and a second sub-component;

determining that a particular area of the first sub-component will be in physical communication with the second sub-component; and

associating the interface with the particular area.

3. The method as recited in claim 2, wherein the third party feature comprises a particular carpentry joint.

4. The method as in claim 3, wherein the particular carpentry joint comprises a joint selected from the group consisting of a bridle joint, a scarf joint, a butt joint, a miter joint, a lap joint, a box joint, a dovetail joint, a dado joint, a groove joint, a tongue and groove joint, a mortise and tenon joint, a bridsmouth joint, a finger joint, a halved joint, and a splice joint.

5. The method as recited in claim 1, wherein identifying the interface comprises:

identifying a first sub-component;

determining that the first sub-component will be in physical communication with a hardware component; and

associating the interface with the first sub-component.

6. The method as recited in claim 5, wherein the third party feature comprises a particular hardware component.

7. The method as recited in claim 1, further comprising:
 - receiving a first configuration file, wherein the first configuration file comprises a first listing of third-party features that are associated with a particular manufacturer; and
 - automatically determining that the interface is associated with a specific third-party feature listed in the first configuration file.
8. The method as recited in claim 7, further comprising:
 - receiving a second configuration file, wherein the second configuration file comprises a different listing of third party features that are associated with another manufacturer; and
 - automatically determining that the interface is associated with a specific third-party feature listed in the second configuration file.
9. The method as recited in claim 1, further comprising:
 - automatically resolving anomalies created by the incorporation of the third party feature.
10. The method as recited in claim 9, wherein automatically resolving anomalies comprises:
 - adjusting a size of the at least one sub-component of the architectural design to accommodate the third party feature.

11. A computer system for incorporating third party components into an architectural design, comprising:

one or more processors; and

one or more computer-readable media having stored thereon executable instructions that when executed by the one or more processors configure the computer system to perform at least the following:

generate within a computer aided drafting program an architectural design for an architectural element, wherein the architectural design comprises one or more sub-components, each sub-components within the one or more sub-components is associated with a different independently executable software objects, each independently executable software object comprising independent variables and functions that are specific to different architectural elements that are represented by each sub-component;

identify at least one interface for manufacturer specific components within the architectural element, wherein the at least one interface is associated with a particular independently executable software object that comprises independent variables and functions that are specific to a particular architectural element that comprises the at least one interface, wherein the particular independently executable software object inherited its characteristics from a parent independently executable software object from which the particular independently executable software object was split;

receive a configuration list, wherein the configuration list comprises manufacturer specific components and manufacturer specific specifications for a particular manufacturer;

automatically identify a manufacturer specific component and manufacturer specific specifications from the configuration list that correspond with the at least one interface; and

automatically incorporate the manufacturer specific component into the architectural design, such that the design represents a mixture of user design format and formatting from the incorporated manufacturer-specific component and specifications, wherein the independently executable software object provides direction about the correct placement of the manufacturer specific component.

12. The system as recited in claim 11, wherein automatically incorporating the manufacturer specific components into the architectural design, comprises:

determining that the manufacturer specific component requires a third party feature to be incorporated within the architectural design; and

automatically adjusting the architectural design to incorporate the third party feature.

13. The system as recited in claim 12, wherein the executable instructions include instructions that are executable to configure the computer system to:

automatically incorporate the third party feature into a computer numerical control file.

14. The system as recited in claim 13, wherein the third party feature comprises a specific carpentry joint.

15. The system as recited in claim 13, wherein the third party feature requires a specific millwork operation.

16. The system as recited in claim 11, wherein the executable instructions include instructions that are executable to configure the computer system to:

automatically resolve an inconsistency that is generated by the incorporation of the manufacturer specific component into the architectural design.

17. The system as recited in claim 16, wherein automatically resolving an inconsistency comprises:

identifying that the manufacturer specific component is not properly sized to fit within the at least one interface for the manufacturer specific component; and

automatically adjusting the architectural design to properly incorporate the manufacturer specific component.

18. The system as recited in claim 16, wherein automatically resolving an inconsistency comprises:

identifying that the manufacturer specific component comprises a specific type of carpentry joint;

determining that the incorporation of the specific type of carpentry joint creates an anomaly within the architectural design; and

automatically adjusting the architectural design to resolve the anomaly.

19. The system as recited in claim 11, wherein the same architectural design is used independently by a first manufacturer and a second manufacturer, such that:

the architectural design relies upon a first configuration list to automatically incorporate a first set of manufacturer specific components from the first manufacturer into the architectural design; and

the architectural design relies upon a second configuration list to automatically incorporate a second set of manufacturer specific components from the second manufacturer into the architectural design.

20. A computer-based system comprising one or more processors configured to execute computer-executable instructions for a method for incorporating third party features into an architectural design, the method comprising:

generating an architectural design comprising one or more sub-components, wherein each of the one or more sub-components is associated with a different independently executable software object that is unique to each sub-component selected from the one or more sub-components and each different independently executable software object comprises independent variables and functions that are specific to different architectural elements that are represented by each sub-component selected from the one or more sub-components;

identifying a hardware interface of a third party feature with at least one sub-component of the architectural design, wherein the hardware interface is configured to receive the third party feature;

receiving specifications of the third party feature;

automatically incorporating the third party feature into the interface, wherein:

a particular independently executable software object that is unique to the at least one sub-component of the architectural design provides direction about the correct placement of the third party feature with respect to the hardware interface, wherein the particular independently executable software object inherited its characteristics from a parent independently executable software object from which the particular independently executable software object was split; and

automatically resolving anomalies that the incorporation of the third party feature creates.