

ASSOCIATING COMPUTER-EXECUTABLE OBJECTS WITH TIMBER
FRAMES WITHIN AN ARCHITECTURAL DESIGN ENVIRONMENT

BACKGROUND OF THE INVENTION

The Field of the Invention

5 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 or building information (BIM) software for designing the aesthetic as well as functional aspects of a given residential or commercial space. For example, a designer might use
20 a CAD or BIM program to design a building or part of a building, and then utilize drawings or other information from that program to order or manufacture building components.

The design and build of buildings or other architectural elements such as mezzanines can be an expensive and time consuming process, and conventional
25 automation of such designs is minimal. For example, conventional systems may require a building to be meticulously designed within a CAD or BIM program. Further, prior to designing the building or architectural element within the CAD or BIM program, conventional systems may require that the specifications of the end

product be exactly known ahead of time. As such a designer may need to know the exact dimensions of the objects being designed, along with all of the building features such as wall type and thickness, building materials, door placement, etc. Conventional downstream manufacturing processes are then typically performed through manual
5 interpretation of drawings and manual input into manufacturing systems.

In at least some conventional systems, a designer or manufacturer may incur great expense 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
10 account the correct measurement. In addition, large cost can be incurred by simply switching from one provider to another. For example, different construction methods 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 of the CAD design.

15 Accordingly, there are a number of problems in the art that can be addressed.

BRIEF SUMMARY OF THE INVENTION

Implementations of the present invention overcome one or more problems in the art with systems, methods, and computer program products for automatically incorporating a timber framework into an architectural design. For example, in at
20 least one implementation of the present invention, as a designer designs and manipulates an architectural design, an independently executable software object can provide corresponding characteristics that are automatically incorporated into a timber framework within the architectural design. Similarly, in at least one implementation, the software can leverage multiple independently executable
25 software objects that are associated with the architectural design to automatically configure a timber framework to incorporate architectural component specific features. One will appreciate that this can provide a number of advantages to designers incorporating a timber framework within a design.

For example, a method in accordance with at least one implementation for
30 creating a timber framework within a variable spatial framework can include receiving a variable spatial framework for use in designing and manufacturing an architectural component. The variable spatial framework may digitally define the

architectural component. The method can further include receiving input to divide the variable spatial framework into one or more independent spaces within the variable spatial framework. Upon receiving input to divide the variable spatial framework, the system can create at least a first independent space and a second independent space
5 based upon the received input. Creating the first independent space and the second independent space can also include associating with at least the first independent space a first independently executable software object. Additionally, the method can include automatically incorporating a digital representation of a first timber framework into the first independent space. The first independently executable
10 software object can determine the characteristics of the first timber framework.

In an additional or alternative implementation, a method can include receiving a variable spatial framework for use in designing and manufacturing an architectural component. The variable spatial framework can digitally define the architectural component. In addition, the method can include receiving input to divide the variable
15 spatial framework into one or more independent spaces within the variable spatial framework. Upon receiving input to divide the variable spatial framework, the system can then perform a step of creating at least a first independent space and a second independent space based upon the received input. Creating the first independent space and the second independent space can include associating with the first independent
20 space a first independently executable software object and associating with the second independent space a second independently executable software object. Additionally, the method can include automatically incorporating a digital representation of a first timber framework into the first independent space. In this case, both the first independently executable software object and the second independently executable
25 software object can at least partly determine the structure and characteristics of the timber framework.

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

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 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:

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

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 2D depicts a planar view of a spatial framework created in accordance with an implementation of the present invention;

Figures 3A illustrates another spatial framework in accordance with the inventive system illustrated in Figure 1;

Figure 3B illustrates the spatial framework of Figure 3A, after a designer has inserted two cube splitters in accordance with an implementation of the present invention;

Figure 3C depicts the spatial framework of Figure 3B upon implementation of a facet splitters in accordance with an implementation of the present invention;

Figure 3D depicts an implementation of a timber framework incorporated into the spatial framework of Figure 3C;

Figure 3E depicts an implementation of a timber framework incorporated of Figure 3C with additional features incorporated;

5 Figure 4 depicts a flowchart of a sequence of acts in a method for incorporating a timber framework into an architectural design in accordance with at least one implementation of the present invention; and

Figure 5 depicts a flowchart of an additional or alternative method for incorporating a timber framework into an architectural design in accordance with at
10 least one implementation of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Implementations of the present invention extend to systems, methods, and computer program products for automatically incorporating a timber framework into an architectural design. For example, in at least one implementation of the present
15 invention, as a designer designs and manipulates an architectural design, an independently executable software object can provide corresponding characteristics that are automatically incorporated into a timber framework within the architectural design. Similarly, in at least one implementation, the software can leverage multiple independently executable software objects that are associated with the architectural
20 design to automatically configure a timber framework to incorporate architectural component specific features. One will appreciate that this can provide a number of advantages to designers incorporating a timber framework within a design.

For example, in at least one implementation of the present invention, a designer can use an object oriented timber frame program to create a spatial
25 framework representative of an architectural component, for example, a timber frame mezzanine. As the user is creating the framework of the mezzanine, the timber frame program can automatically create the timber framework that will make up the architectural component, such as by analyzing the user's input. In particular, in at least one implementation, one or more spaces within the spatial framework can be
30 associated with independently executable software objects. The software objects can automatically determine the characteristics of the timber framework that will allow for structural stability and provide for the features that the user specified when

creating the architectural component.

Additionally, in at least one implementation of the present invention, a designer can create an architectural structure that comprises multiple rooms and/or levels. In particular, various spaces within the architectural structure can be associated with independently executable software objects. The software objects can be mutually aware of each other, and can track the required features and interactions of each respective space. As such, in at least one implementation, the independently executable software objects can automatically adjust based upon input received from a designer.

10 Additionally, in at least one implementation, one or more modules in the timber frame program can identify and track space by assigning an independently executable software object to the space. Using 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
15 the timber framework that defines and partially encloses the space in response to manufacturing specifications and manufacturer specific components.

 Additionally, in at least one implementation, one or more components or modules in the system can automatically incorporate specific features and specifications from a third party manufacturer. In many cases, for example, different
20 manufacturers may use different hardware, different types of joints, and various other different manufacturing elements that each may require unique adjustments to the timber framework. For example, when manufacturing a house, a first facility may use walls of a specific thickness, while a second facility may use walls of a different thickness. In at least one implementation, the architectural design can automatically
25 adjust the timber framework to incorporate either size of wall. 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.

 Implementations of the present invention can also allow a user to create a timber framework for an architectural element without knowing many of the end
30 features that will be incorporated into the architectural element.

 For example, in at least one implementation, a user can create a timber framework for an entire building without knowing the material that will be used, the

dimensions of the resulting building, the specific internal walls, and many other similar details. Once the timber framework is completed, a specific manufacturing facility can enter in the details of its materials, components, configurations, and the actual specifications and/or dimensions of the end product, and the timber framework
5 can automatically adjust to conform to the entered information.

Turning now to the Figures, Figure 1 depicts an architectural schematic diagram of a computer system for designing and manufacturing an architectural element. In particular, Figure 1 shows that an exemplary computer terminal 110 can be in communication with a timber frame software application 100. The timber frame
10 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. One will appreciate that the computer terminal 110 can comprise any computerized device that enables execution of computerized instructions, including any desktop, laptop, or mobile computing device.

Figure 1 further shows that the timber frame software application 100 can comprise a plurality of modules 120, 130, 140, 141, 142, 144, 146, 148, 160 that are adapted to aid in designing a file for timber. For example, Figure 1 depicts an implementation in which the timber frame software application 100 comprises a user interface module 120, a manufacturing preparation module 130, a framework module
20 140, a timber framework module 141, 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 150. One will understand, however, that separating modules into discrete units is at least somewhat arbitrary and that modules can 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, 141,
25 142, 144, 146, 148, 160 of Figure 1 are only shown for illustrative and exemplary purposes of at least one implementation.

In addition, Figure 1 shows that the user interface module 120 can communicate with the computer terminal 110 through a series of data packets 112.
30 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, the user interface module 120 can communicate to – and receive instructions from – the framework module 140. Figure 1 further shows that the framework module 140 can, in turn, communicate with the timber framework module 141, 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 manufacturing facility. Additionally, Figure 1 shows that the various modules 141-148 can communicate with a storage device 150. The storage device 150 can contain, among other things, templates for a variety of different designs, completed designs that can be used on a standalone basis, or designs that can be incorporated into other designs. Additionally, the storage device 150 can also contain tool lists and/or manufacturing information specific to particular manufacturing facilities, and/or to particular design features.

One will appreciate in view of the specification and claims herein that the user interface module 120 provides to the user an option to create and make design changes to a framework 200. In at least one implementation for example, 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 mezzanine for production at a timber frame manufacturing facility. Accordingly, a user may enter instructions into the computer terminal 110 to design and create the mezzanine. 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 then communicate with the appropriate module to execute the request. For example, if the user desires to split the upper surface of the mezzanine into two portions, the framework module 140 communicates with the facet module 142, which can modify and track surfaces within the framework 200. In contrast, and as will be understood more fully herein, if the user desires to split a space into two spaces, the framework module 140 can alternatively use the spaces module 144, which can modify and track spaces within the framework 200. In either case, the framework module 140 can also communicate with the timber framework module 141, which can

automatically configure a timber framework that conforms to the user's input.

For instance, the spaces module 144 can allow the user to split the framework 200 in half and create one half of the mezzanine that might be an open space plan with no walls between the timbers, and another half that is closed in with walls
5 between the timbers. Additionally, a user may use the spaces module 144 to split the framework 200 into any number of other divisions, for example thirds. Of course, one will appreciate that the divisions do not need to be proportional, or equal. For instance, the spaces module 144 can allow a user to move the single split mentioned above such that the open space will only take up one-third of the framework, while
10 the walled space takes up the remaining two-thirds. Based upon the user's input, the timber framework module 141 can create the appropriate timber framework for a mezzanine with the appropriate number of columns and walls.

In at least one implementation, the facet module 142 can perform similar functions on surfaces within the framework 200. For example, the facet module 142
15 can be used to create a split in the front plane of the mezzanine to create large double doors for that space. Similar to the spaces module 144, the facet module 142 can also be used to create split surfaces of different sizes. The timber framework module 141 can then incorporate an appropriate timber framework into the design to allow for a room, or other facet created feature.

20 In addition, the framework module 140 can use the boundary module 146 to automatically check joints within the framework 200 (Figures 2A-2B) to determine if any "anomalies" exist. For example, if a user specifies that a particular joint should be a miter joint, the boundary module 146 can analyze all of the joints within the mezzanine to determine whether the remaining boundaries properly form around the
25 entire mezzanine. 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 mezzanine.

Once the user has finished designing the architectural element (in this case a
30 mezzanine), the manufacturing preparation module 130 can receive manufacturing facility specific details from the configuration list module 160, and can receive specifications relating to the final details of the architectural element. For example, a

particular manufacturing facility may use a dovetail joint to assemble the timber frame components. Additionally, the 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
5 communicate instructions to the appropriate modules 141, 142, 144, 146 to automatically adjust the framework. For example, upon receiving instructions from the manufacturing preparation module 130, the timber framework module 141 can automatically adjust the timber framework of the mezzanine to create a design that incorporates the dovetail joint, wood type and wood thickness, and that can be
10 manufactured at the 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).

In at least one implementation, the manufacturing preparation module 130
15 can also automatically prepare the same framework 200 to be manufactured in any number of different facilities, even though each facility may have specific manufacturing requirements. Specifically, once a particular framework 200 has been designed, the framework 200 can be used at a number of different manufacturing facilities as long as the configuration list module 160 has access to a
20 configuration list 152 that is associated with the specific facility. The configuration list 152 can be stored in the storage device 150, accessed through a network connection, or otherwise made available to the software 100.

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

30 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
5 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
10 changes in a parent object flow through to a child object and vice versa. For example, a software object created for a mezzanine may have several child objects for each column.

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
15 next to the timber object). For example the above-mentioned mezzanine software object and column software objects can independently execute in a correlated fashion to ensure each corresponding physical component (i.e., the mezzanine top floor surface, or the column) is positioned appropriately, or otherwise colored and designed consistent with the user's specifications. For example, a column software object can
20 identify that its location next to a wall renders a physical column unnecessary, and accordingly can automatically incorporate a bracket to attach directly to the wall in place of a physical column.

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
25 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
30 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
5 implementation, the creation of two new spaces 212, 214 also results in the creation of two new independently executable software objects (not shown), which are associated with each space 212, 214. In at least one implementation, a user can specify what particular architectural components that executable software objects are intended to represent (e.g., timber column or beam, wall, door, window.). It
10 should be understood that while cubes and squares are used to illustrate embodiments of the present invention within this application, in at least one implementation, many different shapes and configurations of a framework 200 can be used.

In at least one implementation, the newly created independently executable
15 software objects associated with spaces 212 and 214 may each inherit the parameters and characteristics of the independently executable software object that was originally associated with space 250. Due to this inheritance, if space 250 originally comprises a building with multiple floors, after the split, spaces 212 and 214 can each comprise separate, independently executable software objects that correspondingly comprise
20 apartments that mirror the floor divisions of space 250. In at least one implementation, the independently executable software object that was originally associated with space 250 continues to exist and, in part, functions to manage the two new independently executable software objects that are now associated with the new spaces 212, 214.

25 The cube splitter 210 can also create a new facet 216 within the framework 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 thickness, such that the facets 216, 220, 222 comprise physical surfaces within the architectural element. Similarly, or alternatively, the facets 216,
30 220, 222 can comprise thicknesses of zero, resulting in the facets 216, 220, 222 only being represented within the framework 200 but not within the finished architectural element.

Figure 2C depicts the framework 200 of Figure 2B, albeit comprising two facet splitters 230, 232. As depicted, the facet splitters 230, 232 can split facet 222 (created from splitting facet 204) into three new facets 234, 236, 238. In at least one implementation, a facet splitter 230, 232 is configured to split only facets, such as
5 facets 220, 222, 234, 236, 238. This is as opposed to a cube splitter 210, which can be configured in at least one implementation to split an entire space 250 (i.e., into spaces 212, 214, Figures 2B-2C). 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
10 space 214. In an alternative embodiment, each facet 234, 236, 238 can be associated with its own independently executable software object.

Figure 2D depicts a quasi-two-dimensional view of front face 204 of the framework 200 from Figure 2C, after various space and facet splitting. As mentioned previously, the view is quasi-two-dimensional because each facet can in fact comprise
15 a thickness, when so set by the user. Accordingly, Figure 2D 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 2D 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,
20 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. 2D). 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
25 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.

30 For example, Figure 2D shows that 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 1/3 up the length of cube splitter 210, and similarly, end point 306 being location 1/3 up the length of side boundary 340.

5 One will understand that using a finite length or a proportional length can impact the future manufacture of the architectural element. For example, if a user resizes an item in framework 200 by expanding the framework in all directions, 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
10 other hand, can change in absolute position such that each end point 302, 306 is 1/3 up the length of their respective boundaries 210, 340 (e.g., between points 300/303 and/or 304/307), without regard to the actual length of boundaries 210 and 340.

 In at least one implementation, the framework 200 can be shrunk so much that absolute distance 310 and distance 312 exceed 2/3 of the total length of
15 boundaries 210 and 340 respectively. One will understand that this can cause facet splitter 230 to overlap facet splitter 232. 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 230 or 232, but not both. For example, in at least one implementation, a user can set an option to automatically
20 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 to cube splitter 210 created over subsequent facet splitters 230, 232, or to cube splitter 210. In contrast, a user may be able to set an option that
25 gives priority to the last facet splitter 230, 232, or to cube splitter 210 created over previous facet splitters 230, 232, or to cube splitter 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.

30 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. One will understand that in at least one
5 implementation, the location of facets 220, 234, 236, 238 can impact the timber framework of a particular architectural component. As such, using the above-described methods, a user can exercise significant control over the proportion and layout of an architectural component, and the timber framework module 141 can automatically incorporate a timber framework into the architectural piece in
10 accordance with the user's specifications.

Figures 3A-3E depict various implementations of a timber framework incorporated into the variable spatial frameworks similar to those described in Figures 2A-2D. In particular, Figure 3A depicts a spatial framework 300 that comprises a rectangular cube. In at least one implementation, an independently
15 executable software object that is associated with space 302 determines the timber framework's characteristics and form. One will understand that there are a variety of different shapes and configurations that the spatial framework 300 can take – other than rectangular – and still conform to the timber frame software 100 within this disclosure. In at least one implementation, the independently executable software
20 object automatically determines the particular configuration of a timber framework based upon input received from a user. For example, a user may indicate that a particular spatial framework is being used to design a mezzanine. Accordingly, the independently executable software object can configure the timber framework in a way that is most conducive for a mezzanine.

For example, Figure 3B depicts an implementation of the variable spatial
25 framework 300 from Figure 3A that has been intersected by a first cube splitter 310 and a second cube splitter 312. After the cube splitters 310, 312 are inserted the spaces module 144 can create three new spaces 320, 322, 324 in the place of the original space 302. Additionally, the spaces module can also create three new independently
30 executable software objects (not shown) that are associated with each space 320, 322, 324 respectively. In at least one implementation, the spatial framework 300 can be associated with a scale, such that a designer is able to create spaces of certain

dimensions. For example, the spatial framework may comprise a height that is associated with a desired mezzanine height and a length that is similarly associated with a desired mezzanine length.

Returning to the figures, Figure 3C depicts an implementation of the variable spatial framework of Figure 3B wherein a surface (i.e., a facet) has been bisected by a facet splitter 330. In contrast to the cube splitters 310, 312 that were depicted in Figure 3B, the facet splitter 330 does not completely bisect space 322. Instead, the facet splitter 330 bisects the upper surface of space 322 and creates two facets 332, 334.

After creating the two facets 332, 334, a designer can designate that each of the facets comprise unique attributes. For example, the designer can designate that facet 332 comprise a solid surface of a particular dimension and material such as a constructed wall. The designer can also designate that facet 334 comprise a width zero, which de facto designates facet 334 as open space. Additionally, in creating this exemplary mezzanine the designer can designate that facet 336 comprise a width of zero, and thus in effect remove facet 336 from the final product.

In Figure 3D, the timber framework module 141 has applied a timber framework 326 to the spatial framework 300 of Figure 3C. In at least one implementation of the timber frame software 100, the user interface module 120 can display a digital rendering of the timber framework 326 to a designer. Additionally, a designer may be able to view a timber framework 326 at different points during the design of a spatial framework 300. For example, a designer may wish to view the timber framework 326 throughout the entire design process.

Each space 320, 322, 324 of Figure 3C now comprises a room or area within Figure 3D. Additionally, each space is still associated with an independently executable software object. In at least one implementation, when creating the timber framework 326, the timber framework module 141 relies upon the independently executable software objects to determine the characteristics and parameters for the timber framework 326. For example, the independently executable software objects may comprise information relating to the type of framework (e.g., type of wood, dimensions of wood, placement of wood) that should be used. Additionally, the independently executable software objects may also be mutually configurable. For

example, if a designer designates the independently executable software object associated with space 320 as comprising a pine attribute, one or more of the independently executable software objects can communicate the attribute to each other, such that the entire mezzanine structure is pine. Similarly, in at least one
5 implementation, if a designer adjusts an attribute within space 322, the independently executable software objects can automatically make adjustments to ensure that the mezzanine is structurally sound. In at least one implementation, the independently executable software object can make these various changes by passing data among themselves and by performing various calculations.

10 Accordingly, once a timber framework 326 has been created, or during the design of a timber framework 326, a user can provide material constraints or specifications to the independently executable software objects, that cause the timber framework 326 to automatically adjust to conform to the new constraints. In at least one implementation, the independently executable software objects also allow a
15 spatial framework 300 to be dynamically adjustable without requiring significant redesign work. For example, in at least one implementation, the dimensions of the mezzanine can be expanded or shrunk, and the independently executable software objects can automatically make the necessary changes to the timber framework 326.

Figure 3E depicts the mezzanine of Figure 3C with additional features and
20 adornments. For example, each of the rooms comprises one or more walls 346 and one or more wall attachments 344. In at least one implementation, the configuration of the walls is determined, at least in part, by the independently executable software object. For example, the independently executable software object can comprise directions that specific walls should be of a glass material, while other walls should
25 comprise wood.

Additionally, in at least one implementation, the independently executable software objects can also allow for major features, such as stairs 340, to be added to a timber framework 326. For example, the independently executable software objects may coordinate the proper placement, structure, and design of a set of stairs 340. In at
30 least one implementation, a designer can pick a particular location and configuration of a set of stairs 340 and the independently executable software object can make the necessary additional configurations for the stairs to match the mezzanine and to be

safe.

In at least one implementation, a designer can designate that the mezzanine comprises an upper floor. The independently executable software objects can make the necessary adjustments to the timber framework 326 to ensure that the upper floor
5 is structurally sound. Additionally, in at least one implementation, one or more of the independently executable software objects can configure a railing 342 that is positioned around the top of the mezzanine. For example, an independently executable software object can adjust an attached railing 342 so that the railing matches the style of the space 320, 322, 324 with which the independently executable
10 software object is associated.

Additionally, in at least one implementation, the timber framework software
100 can automatically incorporate a variety of third party features into a timber framework. For example, various different railings 342 can be automatically incorporated into the timber framework. In at least one implementation, one or more
15 independently executable software objects can automatically adjust and manipulate the timber framework to allow for the addition of a particular railing.

In at least one implementation, each independently executable software object is primarily responsible for the configuration of a timber framework 326 that at least partially encloses the independently executable software object's respective space.
20 For example, the timber framework 326 depicted into Figure 3E comprises three spaces 320, 322, 324. The independently executable software object that is associated with space 320 may be primarily responsible for the configuration of the timber framework that at least partially surrounds space 320. Similarly, the independently executable software objects associated with space 322, 324 may be primarily
25 responsible for the configuration of the timber frameworks that at least partially surround the respective spaces 322, 324.

In at least one implementation, independently executable software objects that are associated with adjacent spaces can create cooperative features. For example, Figure 3D depicts adjacent spaces 320 and 322, which share a common wall 328. In
30 at least one implementation, the independently executable software objects that are associated with spaces 320 and 322 can cooperatively define the structure and attributes of the shared wall 328. For example, in Figure 3D the independently

executable software objects created a two pillars that both supports the mezzanine upper floor and divides the two spaces 320, 322.

In addition to creating cooperative features, independently executable software objects can also communicate structural information to various other independently executable software object dispersed throughout a timber framework 326. For example, in at least one implementation the spaces 320, 322, 324 can communicate and adjust to ensure the structural stability of the entire mezzanine structure, including the upper floor.

Accordingly, Figures 1-3E and the corresponding text illustrate or otherwise describe one or more components, modules, and/or mechanisms for automatically incorporating timber framework into a variable spatial framework. 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 4 and 5 and the corresponding text illustrate or otherwise describe a sequence of acts in a method for automatically incorporating timber framework within an architectural design. The acts of Figures 4 and 5 are described below with reference to the components and modules illustrated in Figures 1-3E.

For example, Figure 4 illustrates that a method for automatically incorporating a timber framework within architectural design can comprise an act 400 of receiving a variable spatial framework. Act 400 includes receiving a variable spatial framework for use in designing and manufacturing an architectural component, wherein the variable spatial framework digitally defines the architectural component. For example, in Figure 2C and the accompanying description, the timber frame software 100 receives a variable spatial framework that may be the initial design of a furniture piece, a room, a house, a mezzanine, or some other similar architectural structure or component.

Figure 4 also shows that the method can comprise an act 410 of dividing the variable spatial framework into spaces. Act 410 includes receiving input to divide the variable spatial framework into one or more independent spaces within the variable spatial framework. For example, Figure 2B depicts a variable spatial framework that has been divided with a cube splitter 210. After the variable spatial framework is divided, two spaces 212, 214 are disposed within the variable spatial framework.

Additionally, Figure 4 also shows that the method can comprise an act 420 of creating a first and second space. Act 420 includes creating at least a first independent space and a second independent space based upon the received input, wherein creating the first independent space and the second independent space includes associating with
5 at least the first independent space a first independently executable software object. For example, Figure 2B depicts a variable spatial framework that has been divided into two spaces 212, 214 with a cube splitter 210. After the two spaces are created, as described above, an independently executable software object can be associated with at least the first space.

10 Further, Figure 4 shows that the method can comprise an act 430 of automatically incorporating a timber framework into the first space. Act 430 can include automatically incorporating a digital representation of a first timber framework into the first independent space, wherein the characteristics of the first timber framework are determined by the first independently executable software
15 object. For example, in Figure 3D and the accompanying description describe a timber framework 326 is automatically incorporated into the spatial framework of Figure 3C. As is further described above, an independently executable software object that is associated with the space 320, at least in part, determines characteristics relating to the timber framework that at least partially encloses the
20 space 320. For example, the independently executable software object can determine, among other things, the type of wood framework, the type and location of fasteners, the thickness of specifications of the wood framework, and the general structure of the wood framework.

In addition to the foregoing, Figure 5 illustrates that an additional or
25 alternative method of automatically incorporating a timber framework within an architectural design can comprise an act 500 of associating a first software object with a first space within a variable framework. Act 500 includes associating a first independently executable software object with a first independent space within a variable spatial framework. For example, Figure 2B depicts a variable spatial
30 framework that has been divided into two spaces 212, 214 with a cube splitter 210. After the two spaces are created, as described above, an independently executable software object can be associated with at least the first space.

Figure 5 also shows that the method can comprise an act 510 of associating a second software object with a second space within a variable framework. Act 510 includes associating a second independently executable software object with a second independent space within the variable spatial framework. In at least one implementation, the first independent space and the second independent space are adjacent to each other within the variable spatial framework. For example, Figures 2B-2C depicts a variable spatial framework that has been divided with a cube splitter 210. In addition, Figures 2B-2C show that after the variable spatial framework is divided, two spaces 212, 214 are disposed within the variable spatial framework. Additionally, it is described within the accompanying description that an independently executable software object can be associated with each space 212, 214. [0078] Further, Figure 5 also shows that the system can comprise an act 520 of incorporating a digital representation of a timber framework into the variable spatial framework. Act 520 includes automatically incorporating a digital representation of a timber framework into the variable spatial framework. In at least one implementation, the structure and characteristics of the timber framework are determined by at least both the first independently executable software object and the second independently executable software object. For example, in Figure 3D and the accompanying description describe a timber framework 326 is automatically incorporated into the spatial framework of Figure 3C. As is further described above, an independently executable software object that is associated with the space 320, at least in part, determines characteristics relating to the timber framework that at least partially encloses the space 320. For example, the independently executable software object can determine, among other things, the type of wood framework, the type and location of fasteners, the thickness of specifications of the wood framework, and the general structure of the wood framework.

Accordingly, Figures 1-5 and the corresponding text illustrate or otherwise describe a number of components, schematics, and mechanisms for automatically incorporating a timber framework within an architectural design. One will appreciate that these components and modules in accordance with implementations of the invention can allow a designer to develop an architectural element, such as a desk, an entire kitchen, a house, a mezzanine, or any timber frame architectural structure

without knowing the specific timber framework that will ultimately be created. For example, a user can design a kitchen, including cabinets, drawers, counter tops, sink locations, etc., without knowing the location or structure of the underlying timber framework. Similarly, a user can design a mezzanine without knowing the final
5 dimensions of the mezzanine, material of construction, wall and door placement, and other similar attributes.

Embodiments of the present invention may comprise or utilize a special-purpose or general-purpose computer system that includes computer hardware, such as, for example, one or more processors and system memory, as discussed in greater
10 detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general-purpose or special-purpose computer system. Computer-readable media that store computer-executable
15 instructions and/or data structures are computer storage media. Computer-readable media that carry computer-executable instructions and/or data structures are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media and transmission media.

20 Computer storage media are physical storage media that store computer-executable instructions and/or data structures. Physical storage media include computer hardware, such as RAM, ROM, EEPROM, solid state drives (“SSDs”), flash memory, phase-change memory (“PCM”), optical disk storage, magnetic disk storage or other magnetic storage devices, or any other hardware storage device(s)
25 which can be used to store program code in the form of computer-executable instructions or data structures, which can be accessed and executed by a general-purpose or special-purpose computer system to implement the disclosed functionality of the invention.

Transmission media can include a network and/or data links which can be
30 used to carry program code in the form of computer-executable instructions or data structures, and which can be accessed by a general-purpose or special-purpose computer system. A “network” is defined as one or more data links that enable the

transport of electronic data between computer systems and/or modules and/or other electronic devices. 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 system, the computer system may view the
5 connection as transmission media. Combinations of the above should also be included within the scope of computer-readable media.

Further, upon reaching various computer system components, program code in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (or vice versa). For
10 example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a “NIC”), and then eventually transferred to computer system RAM and/or to less volatile computer storage media at a computer system. Thus, it should be understood that computer storage media can be included in computer system components that
15 also (or even primarily) utilize transmission media.

Computer-executable instructions comprise, for example, instructions and data which, when executed at one or more processors, cause a general-purpose computer system, special-purpose computer system, or special-purpose processing device to perform a certain function or group of functions. Computer-executable
20 instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code.

Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, tablets, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data
25 links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. As such, in a distributed system environment, a computer system may include a plurality of constituent computer systems. In a

distributed system environment, program modules may be located in both local and remote memory storage devices.

Those skilled in the art will also appreciate that the invention may be practiced in a cloud computing environment. Cloud computing environments may be distributed, although this is not required. When distributed, cloud computing environments may be distributed internationally within an organization and/or have components possessed across multiple organizations. In this description and the following claims, “cloud computing” is defined as a model for enabling on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). The definition of “cloud computing” is not limited to any of the other numerous advantages that can be obtained from such a model when properly deployed.

A cloud computing model can be composed of various characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service, and so forth. A cloud computing model may also come in the form of various service models such as, for example, Software as a Service (“SaaS”), Platform as a Service (“PaaS”), and Infrastructure as a Service (“IaaS”). The cloud computing model may also be deployed using different deployment models such as private cloud, community cloud, public cloud, hybrid cloud, and so forth.

Some embodiments, such as a cloud computing environment, may comprise a system that includes one or more hosts that are each capable of running one or more virtual machines. During operation, virtual machines emulate an operational computing system, supporting an operating system and perhaps one or more other applications as well. In some embodiments, each host includes a hypervisor that emulates virtual resources for the virtual machines using physical resources that are abstracted from view of the virtual machines. The hypervisor also provides proper isolation between the virtual machines. Thus, from the perspective of any given virtual machine, the hypervisor provides the illusion that the virtual machine is interfacing with a physical resource, even though the virtual machine only interfaces with the appearance (e.g., a virtual resource) of a physical resource. Examples of physical resources including processing capacity, memory, disk space, network bandwidth, media drives, and so forth.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are 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
5 description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

I claim:

1. A method for creating a timber framework within a variable spatial framework for use in designing and manufacturing an architectural component, the method comprising:

5 receiving a variable spatial framework for use in designing and manufacturing an architectural component, wherein the variable spatial framework digitally defines the architectural component;

receiving input to divide the variable spatial framework into one or more independent spaces within the variable spatial framework;

10 creating at least a first independent space and a second independent space based upon the received input, wherein creating the first independent space and the second independent space includes associating with at least the first independent space a first independently executable software object and the second independent space a second independently executable software object;

15 automatically incorporating a digital representation of a first timber framework into the first independent space, wherein the characteristics of the first timber framework are determined by the first independently executable software object;

receiving input to modify one or more characteristics associated with the first independent space;

20 in response to the received input, automatically modifying the second independent space based on the received input, wherein automatically modifying the second independent space includes at least:

the first independently executable software object automatically sending a communication associated with the received input to the second independently executable software object; and

25 in response to receiving the communication, the second independently executable software object automatically modifying one or more characteristics associated with the second independent space;

30 calculating structural integrity requirements for the architectural component, wherein calculating the structural integrity requirements includes accounting for the structural support provided by at least a first timber framework associated with the first independent space and a second timber framework associated with the second independent space; and

based upon the calculated structural integrity requirements, automatically incorporating into the first timber framework and the second timber framework support features.

2. The method as recited in claim 1, further comprising:

5 receiving a user specification defining a characteristic of the architectural component; and
automatically adjusting the first timber framework to account for the received user
specification.

3. The method as recited in claim 2, wherein the first independently executable software
10 object performs at least a portion of the automatically adjusting the first timber framework.

4. The method as recited in claim 2, wherein the user specification comprises a material
type that will be used to construct the object.

15 5. The method as recited in claim 2, wherein the user specification comprises a size
measurement of the object that will be manufactured.

6. The method as recited in claim 2, wherein the user specification comprises a thickness
of a material that will be used to construct the object.

20 7. The method as recited in claim 1, further comprising:
identifying a surface associated with a portion of the spatial framework, wherein the surface is
representative of a physical surface in the architectural component; and
receiving input to divide the surface, the surface dividing input changing the surface into
25 multiple independent surfaces.

8. The method as recited in claim 7, wherein the surface comprises a thickness that is also
divided by the dividing input.

9. The method as recited in claim 1, further comprising:
detecting the presence of a second timber framework within the second independent space,
which is adjacent to the first independent space.

5 10. The method as recited in claim 9, further comprising:
automatically combining at least a portion of the first timber framework and the second timber
framework.

10 11. The method as recited in claim 10, wherein combining at least a portion of the first
timber framework and the second timber framework comprises:
identifying at least one adjacent feature in the first timber framework and the second timber
framework that is equivalent; and
replacing the identified equivalent features with a single common feature.

15

12. In a computer-based system for automatically creating a timber framework within a variable spatial framework for use in designing and manufacturing an architectural component, the computer-based system comprising one or more processors for receiving and processing instructions, the instructions when executed by the one or more computer processors directing the computer-base
 5 system to perform a method comprising:

associating a first independently executable software object with a first independent space within a variable spatial framework;

associating a second independently executable software object with a second independent space within the variable spatial framework, wherein the first independent space and the second
 10 independent space are adjacent to each other within the variable spatial framework; and

automatically incorporating a digital representation of a timber framework into the variable spatial framework, wherein the structure and characteristics of the timber framework are determined by at least both the first independently executable software object and the second independently executable software object;

15 receiving input to modify one or more characteristics associated with the first independent space;

in response to the received input, automatically modifying the second independent space based on the received input, wherein automatically modifying the second independent space includes at least:

20 the first independently executable software object automatically sending a communication associated with the received input to the second independently executable software object; and

in response to receiving the communication, the second independently executable software object automatically modifying one or more characteristics associated with the
 25 second independent space;

calculating structural integrity requirements for the architectural component, wherein calculating the structural integrity requirements includes accounting for the structural support provided by at least a first timber framework associated with the first independent space and a second timber framework associated with the second independent space; and

30 based upon the calculated structural integrity requirements, automatically incorporating into the first timber framework and the second timber framework support features.

13. The method as recited in claim 12, further comprising:
receiving a user specification defining a characteristic of the architectural component; and
automatically adjusting the timber framework to account for the received material constraint.

5

14. The method as recited in claim 13, wherein automatically adjusting the timber
framework comprises the first independently executable software object incorporating at least
one new feature into the architectural component and providing timber framework for the at least one
new feature.

10

15. The method as recited in claim 12, wherein the first independently executable software
object provides specifications for building a timber framework that at least partially encloses the first
independent space.

15

16. The method as recited in claim 15, wherein the second independently executable
software object provides specifications for building a timber framework that[[are]] at least partially
encloses the second independent space.

20

17. The method as recited in claim 16, wherein the first independent space and the second
independent space share at least a portion of timber framework at a border between the first
independent space and the second independent space.

25

18. The method as recited in claim 12, wherein automatically incorporating the timber
framework comprises the first independently executable software object determining structures and
characteristics of an architectural sub-component framework that will conform with a particular
architectural sub-component.

19. A computer program product for use at a computer system, the computer program product for creating a timber framework within a variable spatial framework for use in designing and manufacturing an architectural component, the computer program product comprising one or more computer storage media having stored thereon computer-executable instructions that, when executed at a processor, cause the computer system to perform the method, including the following:

receiving a variable spatial framework for use in designing and manufacturing an architectural component, wherein the variable spatial framework digitally defines the architectural component;

receiving input to divide the variable spatial framework into one or more independent spaces within the variable spatial framework;

creating at least a first independent space and a second independent space based upon the received input, wherein creating the first independent space and the second independent space includes associating with at least the first independent space a first independently executable software object and the second independent space a second independently executable software object; and

automatically incorporating a digital representation of a first timber framework into the first independent space, wherein the characteristics of the first timber framework are determined by the first independently executable software object;

receiving input to modify one or more characteristics associated with the first independent space;

in response to the received input, automatically modifying the second independent space based on the received input, wherein automatically modifying the second independent space includes at least:

the first independently executable software object automatically sending a communication associated with the received input to the second independently executable software object; and

in response to receiving the communication, the second independently executable software object automatically modifying one or more characteristics associated with the second independent space;

calculating structural integrity requirements for the architectural component, wherein calculating the structural integrity requirements includes accounting for the structural support provided by at least a first timber framework associated with the first independent space and a second timber framework associated with the second independent space; and

based upon the calculated structural integrity requirements, automatically incorporating into the first timber framework and the second timber framework support features.