US 20180129757A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2018/0129757 A1

Bowman et al. (43) Pub. Date: May 10, 2018

(54) **ENGINEERING SOFTWARE THAT USES A PSEUDO-SINGLETON DESIGN PATTERN WHICH SUPPORTS ASYNCHRONOUS HIERARCHICAL UPDATES IN SYNCHRONOUS COLLABORATIVE SOFTWARE**

(71) Applicant: **Brigham Young University**, Provo, UT (US)

(72) Inventors: **K Eric Bowman**, Provo, UT (US); **Joshua Coburn**, Orem, UT (US); **C. Greg Jensen**, Provo, UT (US)

(21) Appl. No.: **15/621,919**

(22) Filed: **Jun. 13, 2017**

Related U.S. Application Data

(60) Provisional application No. 62/349,424, filed on Jun. 13, 2016.

Publication Classification

(51) **Int. Cl.**

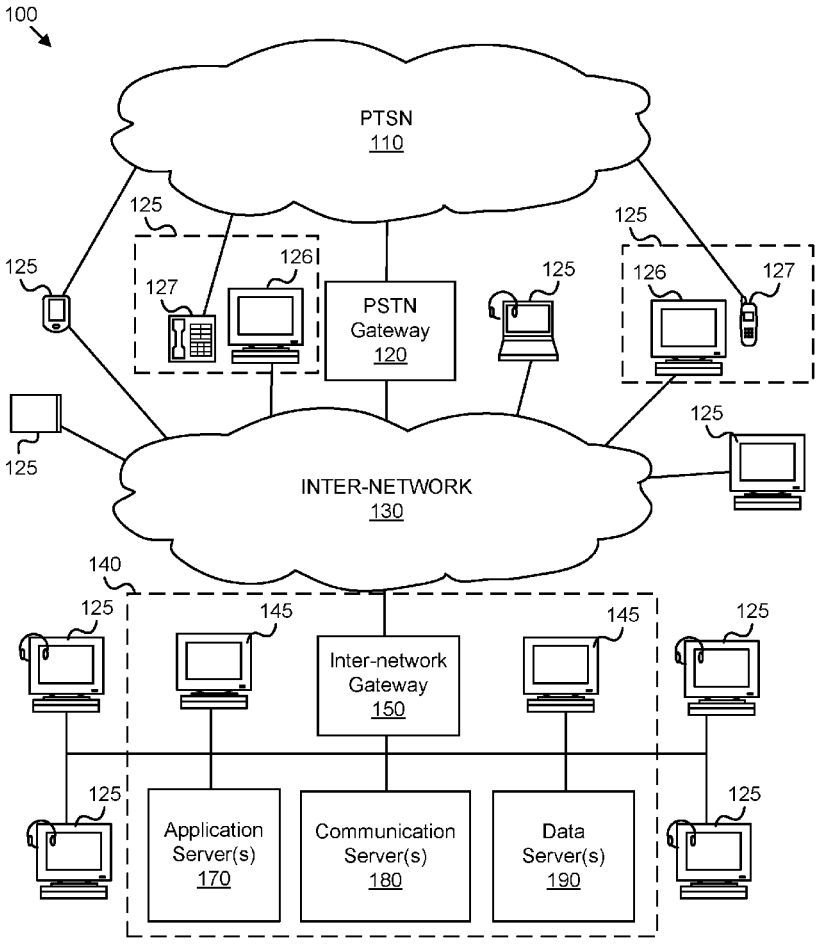
G06F 17/50	(2006.01)
G06F 17/27	(2006.01)
H04L 29/06	(2006.01)
G06F 9/54	(2006.01)

(52) **U.S. Cl.**

CPC ***G06F 17/50*** (2013.01); ***G06F 9/546*** (2013.01); ***H04L 67/42*** (2013.01); ***G06F 17/2735*** (2013.01)

(57) **ABSTRACT**

A method for multi-user CAx editing includes receiving a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment, identifying a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element, comparing the reference portion to a dictionary of the CAx environment, receiving the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary, and creating the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary. A computer program product such as a computer readable medium and a computer system corresponding to the above method are also disclosed herein.



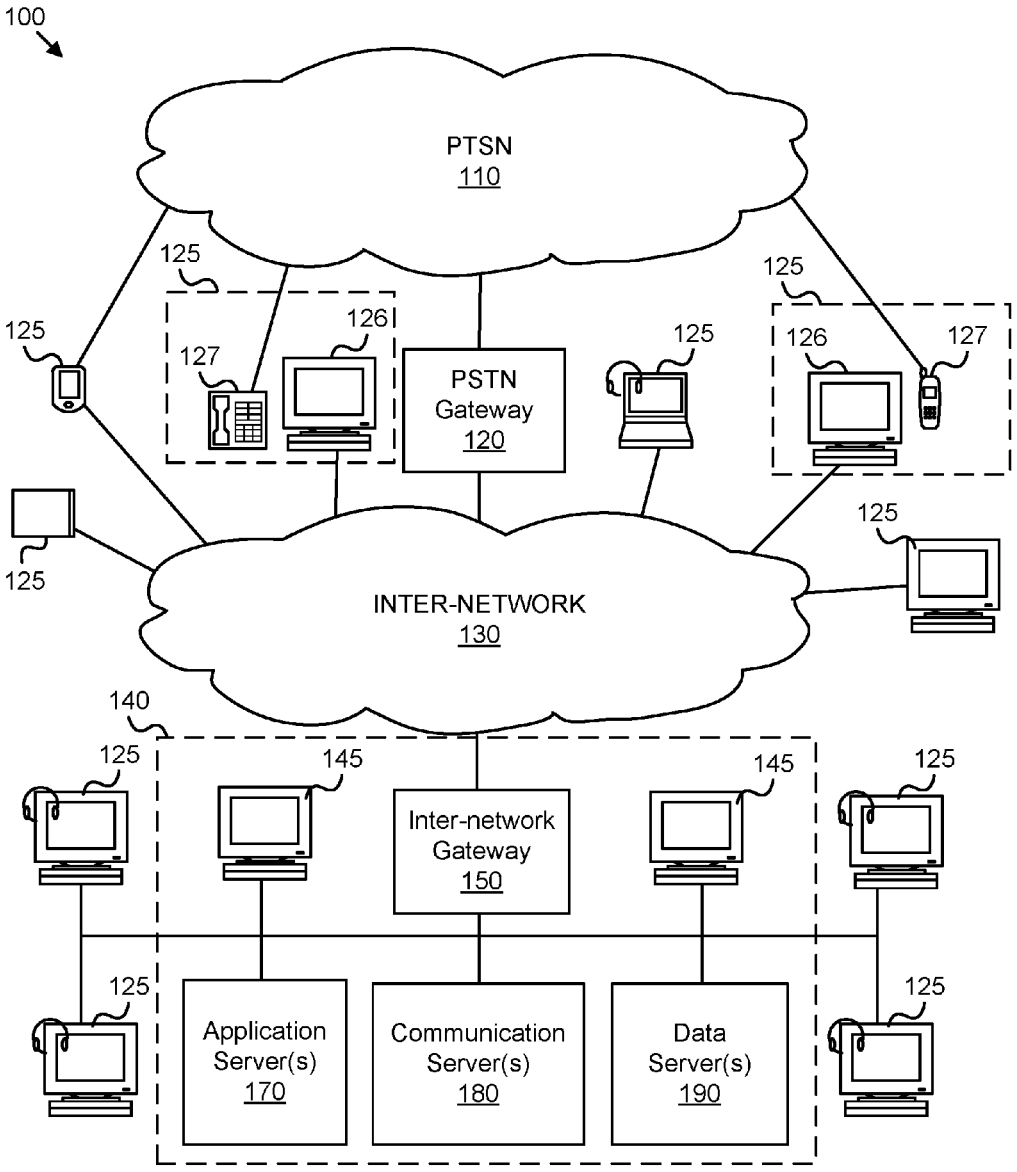

100
PTSN
110
125
126
127
PSTN
Gateway
120
INTER-NETWORK
130
140
145
Inter-network
Gateway
150
Application
Server(s)
170
Communication
Server(s)
180
Data
Server(s)
190

Figure 1

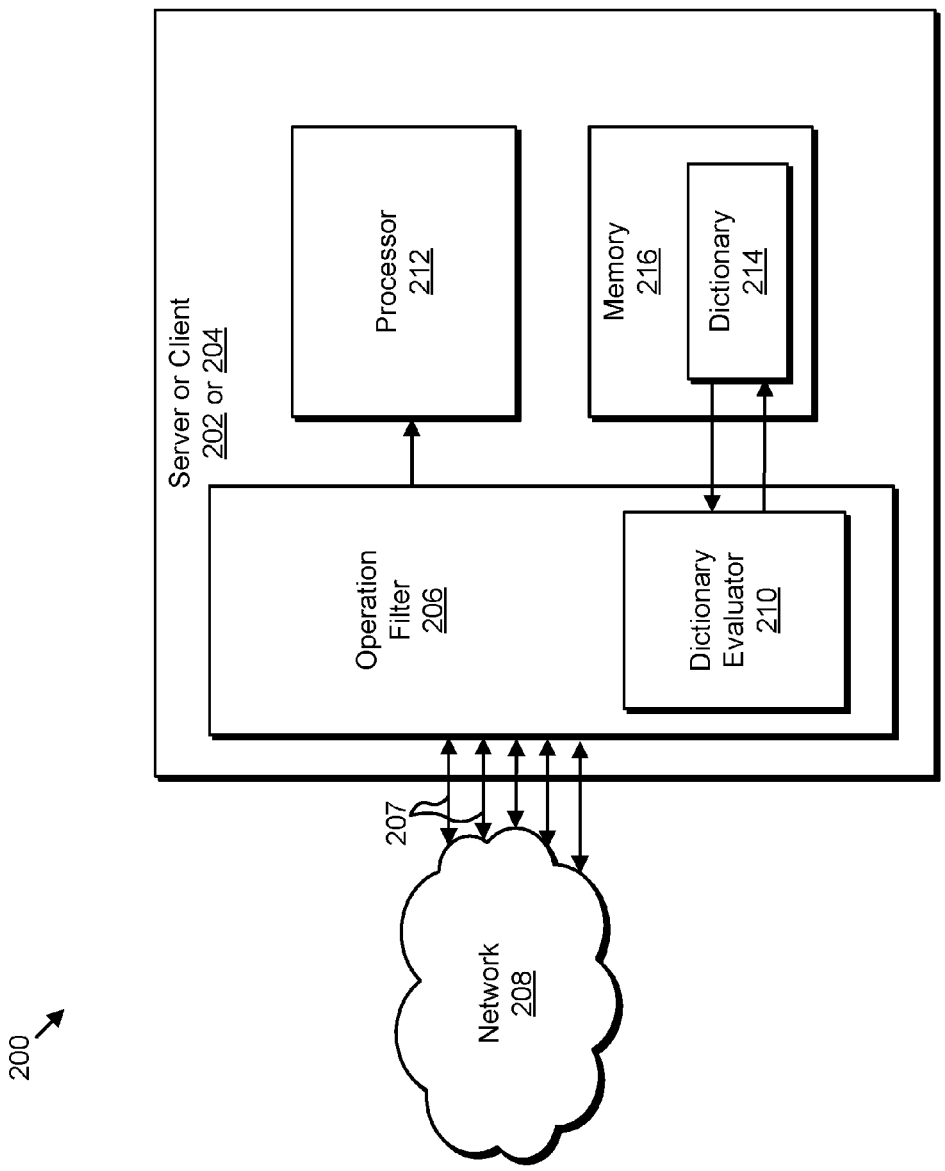

200
Server or Client
202 or 204
Processor
212
Memory
216
Dictionary
214
Operation
Filter
206
Dictionary
Evaluator
210
207
Network
208

Figure 2

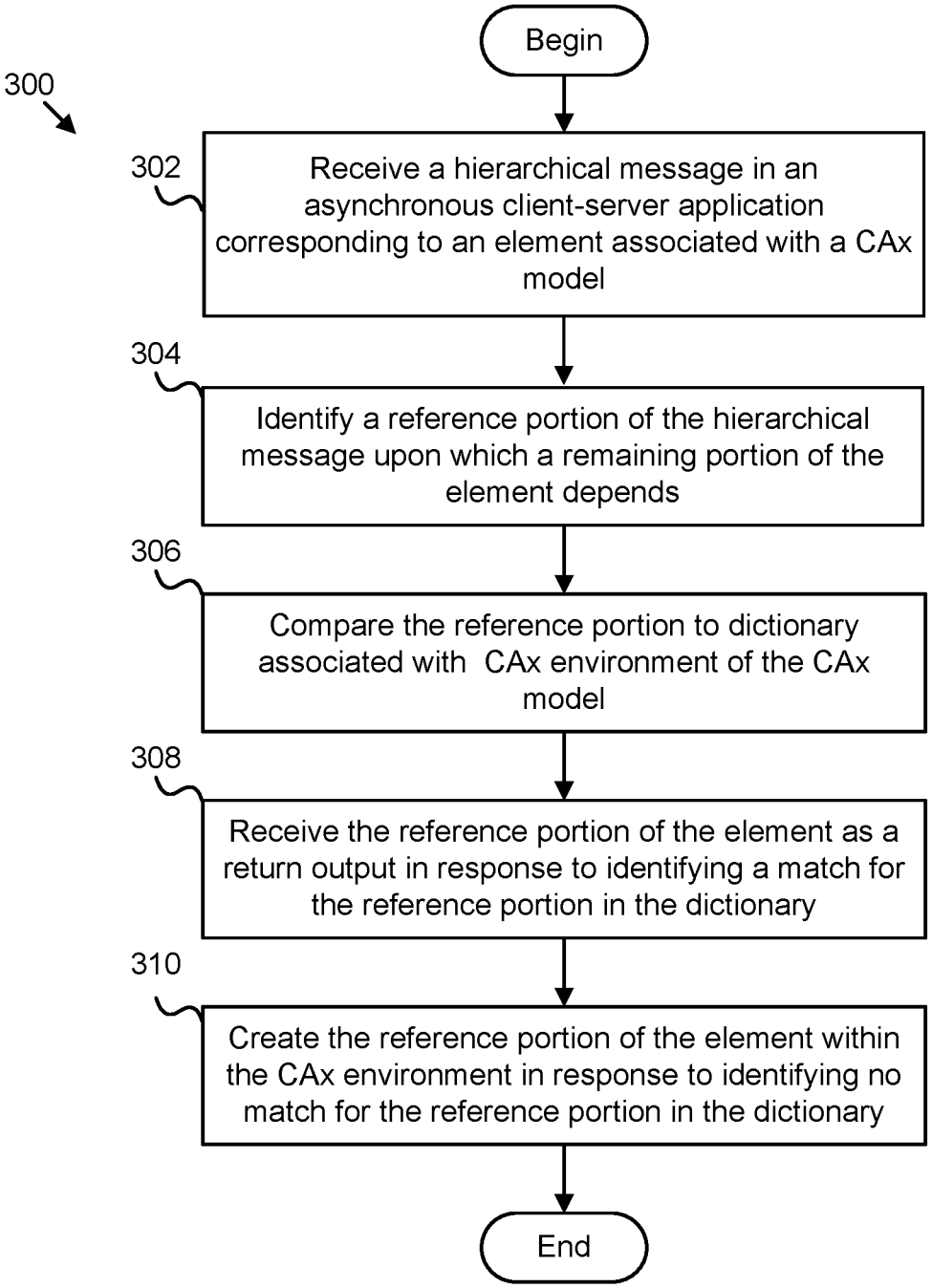

300
Begin
302
Receive a hierarchical message in an asynchronous client-server application corresponding to an element associated with a CAx model
304
Identify a reference portion of the hierarchical message upon which a remaining portion of the element depends
306
Compare the reference portion to dictionary associated with CAx environment of the CAx model
308
Receive the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary
310
Create the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary
End

Figure 3

ENGINEERING SOFTWARE THAT USES A PSEUDO-SINGLETON DESIGN PATTERN WHICH SUPPORTS ASYNCHRONOUS HIERARCHICAL UPDATES IN SYNCHRONOUS COLLABORATIVE SOFTWARE

RELATED APPLICATIONS

[0001] This application claims priority to U.S. provisional application 62/349,424 entitled "Engineering software that uses a pseudo-singleton design pattern which supports asynchronous hierarchical updates in synchronous collaborative software" and filed on 13 Jun. 2016. The above application is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

[0002] The claimed invention relates to computer aided technologies (CAx) such as computer aided design, engineering, analysis, and manufacture in general, and managing data in a CAx environment.

[0003] Multi-user CAx technologies enable multiple users to collaborate on projects. However, if pieces of shared data on operations are interdependent and are received out of order, complex logical checks are needed to determine the order in which the operations are to be applied to determine if one or more of the operations has partially or wholly fulfilled and apply operations, in whole or part, that have not been fulfilled.

SUMMARY OF THE INVENTION

[0004] The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available CAX systems, apparatuses, and methods. Accordingly, the claimed inventions have been developed to provide CAx editing systems, apparatuses, and methods that overcome shortcomings in the art.

[0005] A method for multi-user CAx editing includes receiving a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment, identifying a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element, comparing the reference portion to a dictionary of the CAx environment, receiving the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary, and creating the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

[0006] A computer program product including a computer readable storage medium having program instructions embodied therewith, the program instructions readable/executable by a processor to cause the processor to receive a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment, identify a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element, compare the reference portion to a dictionary of the CAx environment, receive the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary, and create the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

[0007] A computer system including a processor, a memory, and a computer readable medium having instructions encoded thereon to cause the processor to receive a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment, identify a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element, compare the reference portion to a dictionary of the CAx environment, receive the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary, and create the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

[0008] It should be noted that references throughout this specification to features, advantages, or similar language do not imply that all the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

[0009] The described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

[0010] These features and advantages 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

[0011] In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. 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:

[0012] FIG. **1** is a block diagram of one example of a computing and communications infrastructure that is consistent with one or more embodiments of the claimed invention;

[0013] FIG. **2** is a block diagram of one example of a collaborative CAx editing system that is consistent with one or more embodiments of the claimed invention; and

[0014] FIG. **3** is a flowchart diagram of one embodiment of a method for multi-user CAx editing.

DETAILED DESCRIPTION OF THE INVENTION

[0015] Reference throughout this specification to "one embodiment," "an embodiment," or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment, but mean "one or more but not all embodiments" unless expressly specified otherwise. The terms "including," "comprising," "having," and variations thereof mean "including but not limited to" unless expressly specified otherwise. An enumerated listing of items does not imply that any or all of the items are mutually exclusive and/or mutually inclusive, unless expressly specified otherwise. The terms "a," "an," and "the" also refer to "one or more" unless expressly specified otherwise.

[0016] Furthermore, the described features, advantages, and characteristics of the embodiments may be combined in any suitable manner. One skilled in the relevant art will recognize that the embodiments may be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments.

[0017] The present invention may be a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

[0018] The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing.

[0019] A non-exhaustive list of more specific examples of the computer readable storage medium includes the following: a portable computer diskette, a hard disk, a random access memory ("RAM"), a read-only memory ("ROM"), an erasable programmable read-only memory ("EPROM" or Flash memory), a static random access memory ("SRAM"), a portable compact disc read-only memory ("CD-ROM"), a digital versatile disk ("DVD"), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, and any suitable combination of the foregoing. A computer readable storage medium, as used herein, is not to be construed as being transitory signals per se, such as radio waves or other freely propagating electromagnetic waves, electromagnetic waves propagating through a waveguide or other transmission media (e.g., light pulses passing through a fiber-optic cable), or electrical signals transmitted through a wire.

[0020] Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network. The network may comprise copper transmission cables, optical transmission fibers, wireless transmission, routers, firewalls, switches, gateway computers and/or edge servers. A network adapter card or network interface in each computing/processing device receives computer readable program instructions from the network and forwards the computer readable program instructions for storage in a computer readable storage medium within the respective computing/processing device.

[0021] Computer readable program instructions for carrying out operations of the present invention may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Smalltalk, C++ or the like, and conventional procedural programming languages, such as the "C" programming language or similar programming languages.

[0022] The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

[0023] In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

[0024] Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

[0025] These computer readable program instructions may be provided to a processor of a general-purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks. These computer readable program instructions may also be stored in a computer readable storage medium that can direct a computer, a programmable data processing apparatus, and/or other devices to function in a particular manner, such that the computer readable storage medium having instructions stored therein comprises an article of manufacture including

instructions which implement aspects of the function/act specified in the flowchart and/or block diagram block or blocks.

[0026] The computer readable program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other device to cause a series of operational steps to be performed on the computer, other programmable apparatus or device to produce a computer implemented process, such that the instructions which execute on the computer, other programmable apparatus, or other device implement the functions/acts specified in the flowchart and/or block diagram block or blocks.

[0027] The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures.

[0028] For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

[0029] Many of the functional units described in this specification have been labeled as modules to emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

[0030] Modules may also be implemented in software for execution by various types of processors. An identified module of program instructions may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

[0031] The computer program product may be deployed by manually loading directly in the client, server, and proxy computers via loading a computer readable storage medium such as a CD, DVD, etc., the computer program product may be automatically or semi-automatically deployed into a computer system by sending the computer program product to a central server or a group of central servers. The computer program product is then downloaded into the client computers that will execute the computer program product. Alternatively, the computer program product is sent directly to the client system via e-mail. The computer program product is then either detached to a directory or loaded into a directory by a button on the e-mail that executes a program that detaches the computer program product into a directory.

[0032] Another alternative is to send the computer program product directly to a directory on the client computer hard drive. When there are proxy servers, the process will, select the proxy server code, determine on which computers to place the proxy servers' code, transmit the proxy server code, then install the proxy server code on the proxy computer. The computer program product will be transmitted to the proxy server and then it will be stored on the proxy server.

[0033] The computer program product, in one embodiment, may be shared, simultaneously serving multiple customers in a flexible, automated fashion. The computer program product may be standardized, requiring little customization and scalable, providing capacity on demand in a pay-as-you-go model.

[0034] The computer program product may be stored on a shared file system accessible from one or more servers. The computer program product may be executed via transactions that contain data and server processing requests that use Central Processor Unit (CPU) units on the accessed server. CPU units may be units of time such as minutes, seconds, hours on the central processor of the server. Additionally, the accessed server may make requests of other servers that require CPU units. CPU units are an example that represents but one measurement of use. Other measurements of use include but are not limited to network bandwidth, memory usage, storage usage, packet transfers, complete transactions etc.

[0035] When multiple customers use the same computer program product via shared execution, transactions are differentiated by the parameters included in the transactions which identify the unique customer and the type of service for that customer. All the CPU units and other measurements of use that are used for the services for each customer are recorded.

[0036] When the number of transactions to any one server reaches a number that begins to affect the performance of that server, other servers are accessed to increase the capacity and to share the workload. Likewise, when other measurements of use such as network bandwidth, memory usage, storage usage, etc. approach a capacity so as to affect performance, additional network bandwidth, memory usage, storage etc. are added to share the workload.

[0037] The measurements of use used for each service and customer are sent to a collecting server that sums the measurements of use for each customer for each service that was processed anywhere in the network of servers that provide the shared execution of the computer program product. The summed measurements of use units are periodically multiplied by unit costs and the resulting total computer program product service costs are alternatively sent to the customer and or indicated on a web site accessed by the customer which then remits payment to the service provider.

[0038] In one embodiment, the service provider requests payment directly from a customer account at a banking or financial institution. In another embodiment, if the service provider is also a customer of the customer that uses the computer program product, the payment owed to the service

provider is reconciled to the payment owed by the service provider to minimize the transfer of payments.

[0039] The computer program product may be integrated into a client, server, and network environment by providing for the computer program product to coexist with applications, operating systems and network operating systems software and then installing the computer program product on the clients and servers in the environment where the computer program product will function.

[0040] In one embodiment software is identified on the clients and servers including the network operating system where the computer program product will be deployed that are required by the computer program product or that work in conjunction with the computer program product. This includes the network operating system that is software that enhances a basic operating system by adding networking features.

[0041] In one embodiment, software applications and version numbers are identified and compared to the list of software applications and version numbers that have been tested to work with the computer program product. Those software applications that are missing or that do not match the correct version will be upgraded with the correct version numbers. Program instructions that pass parameters from the computer program product to the software applications will be checked to ensure the parameter lists match the parameter lists required by the computer program product.

[0042] Conversely, parameters passed by the software applications to the computer program product will be checked to ensure the parameters match the parameters required by the computer program product. The client and server operating systems including the network operating systems will be identified and compared to the list of operating systems, version numbers and network software that have been tested to work with the computer program product. Those operating systems, version numbers and network software that do not match the list of tested operating systems and version numbers will be upgraded on the clients and servers to the required level.

[0043] In response to determining that the software where the computer program product is to be deployed, is at the correct version level that has been tested to work with the computer program product, the integration is completed by installing the computer program product on the clients and servers.

[0044] The computer program product, in one embodiment, may be deployed, accessed, and executed using a virtual private network (VPN), which is any combination of technologies that can be used to secure a connection through an otherwise unsecured or untrusted network. The use of VPNs is to improve security and for reduced operational costs.

[0045] The VPN makes use of a public network, usually the Internet, to connect remote sites or users together. Instead of using a dedicated, real-world connection such as leased line, the VPN uses "virtual" connections routed through the Internet from the company's private network to the remote site or employee. Access to the software via a VPN can be provided as a service by specifically constructing the VPN for purposes of delivery or execution of the computer program product (i.e. the software resides elsewhere) wherein the lifetime of the VPN is limited to a given period of time or a given number of deployments based on an amount paid.

[0046] The computer program product may be deployed, accessed, and executed through either a remote-access or a site-to-site VPN. When using the remote-access VPNs the computer program product is deployed, accessed, and executed via the secure, encrypted connections between a company's private network and remote users through a third-party service provider. The enterprise service provider (ESP) sets up a network access server (NAS) and provides the remote users with desktop client software for their computers. The telecommuters can then dial a toll-free number or attach directly via a cable or DSL modem to reach the NAS and use their VPN client software to access the corporate network and to access, download and execute the computer program product.

[0047] When using the site-to-site VPN, the computer program product is deployed, accessed, and executed through the use of dedicated equipment and large-scale encryption that are used to connect a company's multiple fixed sites over a public network such as the Internet.

[0048] The computer program product is transported over the VPN via tunneling which is the process of placing an entire packet within another packet and sending it over a network. The protocol of the outer packet is understood by the network and both points, called tunnel interfaces, where the packet enters and exits the network.

[0049] Furthermore, the described features, structures, or characteristics of the embodiments may be combined in any suitable manner. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments. One skilled in the relevant art will recognize, however, that embodiments may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of an embodiment.

[0050] The description of elements in each figure may refer to elements of proceeding figures. Like numbers refer to like elements in all figures, including alternate embodiments of like elements.

[0051] In a multi-user environment, it is desirable to assign different areas, regions, or geometries for different users to work i.e. a workspace. The assigned workspace may or may not be a contiguous region. Casually updating the model for each user may not be effective, because it is likely a user will accidently cross into another user's workspace or create conflicts with simultaneously executed operations. The result may end up creating chaos and result in a less productive or unworkable multi-user environment. Therefore, it is generally desirable that operations be serialized as they are exchanged between users or servers.

[0052] As used herein the phrase 'engineering object' refers to an electronically modeled object that may be edited by a CAx application or tool and CAx model' refers to the electronic model for that object. CAx applications and tools include, but are not limited to, design tools, meshing tools, simulation tools, visualization tools, analysis tools, manufacture planning tools, and manufacture simulation tools.

[0053] FIG. **1** is a block diagram of one example of a computing and communications infrastructure **100** that is consistent with one or more embodiments of the claimed

invention. As depicted, the infrastructure **100** includes various systems, subsystems, and networks such as a public switched telephone network (PSTN) **110**, a TDM gateway **120** connecting the PSTN to an inter-network **130**, a variety of workstations **125**, a data center **140** with administrative terminals **145**, an inter-network gateway **150** connecting a local area network to the inter-network **130**, and various servers such as application servers **170**, communication servers **180**, and data servers **190**. The infrastructure **100** is one example of components that can be operably interconnected to provide an infrastructure for a collaborative CAx system.

[0054] Each workstation **125** may include a separate computing device **126** and a communications device **127** or the computing device and communications device may be integrated into the workstation **125**. Examples of the communications device **127** include a phone, a VOIP device, an instant messaging device, a texting device, a browsing device, and the like. The computing devices **126** may enable graphical view selection. The communications devices **127** may enable users to communicate with other CAx system users.

[0055] The inter-network **130** may facilitate electronic communications between the various workstations and servers. In one embodiment, the inter-network **130** is the internet. In another embodiment, the inter-network **130** is a virtual private network (VPN).

[0056] Various servers such as blade servers within the data center **140** function cooperatively to facilitate concurrent collaborative editing of CAx models by local and remote users. For example, the application servers **170** may provide one or more CAx applications to the local and remote users. Some users may have the CAx applications installed on their local computing devices **126**.

[0057] In some embodiments, the communication servers **180** facilitate communications between the users through various channels or services such as VOIP services, email services, instant messaging services, short message services, and text messaging services. The workstations **125** may leverage such services for user to user communications via the communication servers **180** or via other available service platforms.

[0058] The data servers **190** or the like may store CAx models of design or engineering objects within various model files or records. The data servers may replicate copies of the models for use by various users. Some users may have a local copy of a model.

[0059] FIG. **2** is a block diagram of one example of a collaborative CAx editing system **200**. The illustrated embodiment includes a server or client **202** or **204**, respectively. In some embodiments, the collaborative CAx editing system **200** is embodied at the server level or at the client level. The description below refers to the client **204**, however, it should be borne in mind that other embodiments operate on the server **202**.

[0060] In the illustrated embodiment, the client **204** includes an operation filter **206**. The operation filter **206** receives hierarchical messages **207** from the network **208**. In some embodiments, the hierarchical messages **207** are remote communications from a server or remote client distinct from the client **204**. In some embodiments, the hierarchical messages **207** include one or more remote operations corresponding to a CAx model or object associated with the client **204** and a remote entity. In some embodiments, the hierarchical messages **207** correspond to operations to be executed on the client **204** to update the engineering object or CAx model based on operations completed relative to the CAx model separate from the client **204**. In some embodiments, the hierarchical messages **207** are asynchronous communications.

[0061] In some embodiments, the hierarchical messages **207** may be delayed by network traffic or latencies. In other embodiments, the hierarchical messages **207** may be asynchronous based on one or more network security protocols or certification or proofing processes. Other embodiments of the hierarchical messages **207** may be asynchronous for other reasons or in response to other functions or processes.

[0062] In some embodiments, the hierarchical messages **207** include parent and child data for an operation or element of the CAx model. In some embodiments, the hierarchical messages **207** are in a neutral format that is not compatible with the client **204**. In some embodiments, the hierarchical messages **207** include data which is not in a specified order or organization which corresponds to an executable order relative to the CAx model.

[0063] In the illustrated embodiment, the hierarchical messages **207** are received by the operation filter **206**. In some embodiments, the operation filter **206** receives the hierarchical messages **207** from a network interface device such as a network interface card or other communication component. In some embodiments, the operation filter **206** is a dedicated hardware element of the client **204**. In other embodiments, the operation filter **206** includes at least one software component such as logic or a learning machine to evaluate the hierarchical messages **207**, for example, the dictionary evaluator **210** described below.

[0064] In some embodiments, the operation filter **206** analyzes the hierarchical messages **207** to determine the nature of the hierarchical messages **207**. In some embodiments, the operation filter **206** provides instructions or other data to a processor **212** of the client **204** for execution.

[0065] In some embodiments, the operation filter **206** identifies data from the hierarchical messages **207** as a reference portion of the hierarchical messages **207**. In other embodiments, the operation filter **206** identifies parent and child data from the hierarchical messages **207**.

[0066] In some embodiments, the operation filter **206** evaluates the hierarchical messages **207** by applying a dictionary evaluator **210**. In some embodiments, the dictionary evaluator **210** includes a singleton or pseudo-singleton pattern. In some embodiments, the dictionary evaluator **210** prevents the formation of duplicate instances of an element in at least one of the dictionary **214**, the CAx model, or the CAx environment.

[0067] In some embodiments, the dictionary evaluator **210** takes, as an input, the reference portion of the hierarchical messages **207**. In some embodiments, the dictionary evaluator **210** compares a reference portion of the hierarchical messages **207** to a dictionary **214**.

[0068] In some embodiments, the dictionary evaluator **210** sends the reference portion back to the operation filter **206** in response to identifying a match for the reference portion in the dictionary **214**. In some embodiments, the dictionary evaluator **210** sends a command to create the reference portion of the element within a CAx environment in response to identifying no match for the reference portion in the dictionary **214**.

[0069] In some embodiments, the command to create the reference portion in the CAx environment includes creating the reference portion within the dictionary **214**. In another embodiment, the command to create the reference portion in the CAx environment includes creating the reference portion at a CAx model within the CAx environment. In some embodiments, such a command is executed by the processor **212**.

[0070] In some embodiments, the dictionary **214** is stored in a local memory **216** of the client **204**. In other embodiments, the dictionary **214** is a central dictionary accessed by a plurality of clients **204**. In some embodiments, the dictionary **214** is a central dictionary **214** located on a device separate to, but accessible by, the client **204**. In some embodiments, the dictionary **214** is a copy that is shared across and updated by a network of devices.

[0071] In some embodiments, the memory **216** is a memory device corresponding to the client **204**. In other embodiments, the memory **216** is a memory device separate from the client **204**. For example, the memory **216** may include a memory device corresponding to at least one of a server, another client, a network attached storage, a cloud memory device, and another memory device.

[0072] FIG. **3** is a flowchart diagram of one embodiment of a method **300** for multi-user CAx editing. At block **302**, the illustrated method **300** includes receiving a hierarchical message in an asynchronous client-server application. In some embodiments, the hierarchical message corresponds to an element of a CAx model in a CAx environment. In some embodiments, the hierarchical message is received at a client. In other embodiments, the hierarchical message is received at a server or other network component.

[0073] At block **304**, the method **300** includes identifying a reference portion of the hierarchical message upon which a remaining portion of the element depends. In some embodiments, the reference portion is unique to the element. In some embodiments, the reference portion includes at least one of parent and child data corresponding to the element. In some embodiments, the hierarchical message has a neutral format. In some embodiments, the hierarchical message includes updates to one or more features of a CAx model on the CAx client.

[0074] Examples of some potential features include the shape, dimensions, composition, material properties and tolerances of an object, the mesh size and required accuracy for simulations, the path and tolerances for a manufacturing tool, and any other attribute that may affect the performance of a product and the manufacture thereof.

[0075] In some embodiments, the CAx model is stored locally. In some embodiments, the CAx client includes a local model datastore that contains local copies of CAx models managed by a global model datastore. In some embodiments, the local and global model datastores coordinate together to provide data coherency between local copies of the CAx models and the global copy. In some embodiments, the global model datastore is a redundant and/or a distributed storage system. In some embodiments, the local copies of the CAx models exchange data, such as hierarchical messages, with one another to provide data coherency.

[0076] In some embodiments, the CAx client includes a user interface. In some embodiments, the user interface indicates reception of the first remote operation to by executed by the first CAx client in the CAx environment. In some embodiments, the user interface does not provide a hierarchical message until the operation filter has evaluated the hierarchical message, as described above. In other embodiments, the user interface does not display the hierarchical message on the CAx client. In some embodiments, a level of detail shown by the user interface is user customizable. In some embodiments, the user interface allows for customization of the level of detail shown.

[0077] In some embodiments, the user interface provides a user with a variety of interface elements that facilitate concurrent collaborative editing. Examples of such interface elements include interfaces elements for displaying a feature tree, defining a partitioning surface or equation, selecting, reserving, assigning, locking and releasing geometries, editing regions and features, specifying a feature constraint, selecting and editing geometries, displaying a list of concurrent users, displaying user identifiers proximate to assigned editing regions, presenting a list or other view of geometries and/or features, prioritizing user access rights and priorities (e.g. by a project leader) including viewing privileges, selecting user-to-user communication channels, initiating communication with another user, and providing access to software tools corresponding to various stages or layers associated with an engineering object. In some embodiments, the user interface module **230** responds to mouse events, keyboard events, and the like.

[0078] In some embodiments, the CAx client includes a user-to-user communication module which facilitates direct communication between different users. In some embodiments, the user-to-user communication module leverages one or more of a variety of communication services such as those detailed above. Communication may be between concurrent users as well as users that may not be actively editing an object. The interface elements provided by the user interface may enable a user to select users or groups as a target for a particular message or ongoing conversation.

[0079] At block **306**, the method **300** includes comparing the reference portion to a dictionary associated with the CAx environment. In some embodiments, comparing the reference portion to the dictionary include the application of a dictionary evaluator having a singleton or pseudo-singleton design pattern. In some embodiments, the design pattern of the dictionary evaluator or the dictionary prevents or reduces the likelihood of creating a duplicate instance of the element associated with the CAx model. Embodiments of coding for such a design is described below.

[0080] At block **308**, the method **300** includes executing the first remote operation on the first CAx client in response to a determination that the first CAx client is idle. In some embodiments, a processor of the first CAx client sends a notification indicating that the first CAx client is idle. In other embodiments, a monitor generates the notification in response to determining that the CAx client is idle.

[0081] In some embodiments, the singleton design pattern ensures that a single copy of a class can be created and no duplicate copies may be created. In some embodiments, if an instance of the class exists, the instance is returned by a singleton pattern or other mechanism. Other embodiments may provide fewer or more restrictions or allow more instances to be created.

[0082] In some embodiments, a constructor of a design pattern is made private and operators are allowed to use the "GetInstance" feature with the corresponding static "Instance" variable. In this manner, the instantiation of a

class is limited to one object or element. In some embodiments, the features and variable are given other names. It is noted that specific naming conventions are not limiting but merely examples.

[0083] In some embodiments, multiple versions of the type, such as a line or other element, may be created but each version is unique. In a CAx environment, multiple lines exist but duplicates of the same line are not useful and can create conflicts and other problems. In some embodiments, a dictionary is organized and maintained to prevent the formation of duplicate elements. In some embodiments, the dictionary can be checked to determine if an element exists or if an element can be created without duplication.

[0084] In one example, a pseudo-singleton function may be associated with a line element. In the above example, the pseudo-singleton function may be applied to a geometric line element. In some embodiments, creation of a line requires a particular order of operations. For example, in some systems, end points must be created prior to establishment of the line itself.

[0085] In some embodiments, it can be difficult to determine, from an asynchronous message corresponding to a line, whether the endpoints of the line have been created. In some embodiments, the line calls a "GetInstance" function based on the end points. As mentioned above, other names and naming conventions may be applied to other embodiments. In some embodiments, a pseudo-singleton design pattern returns the input end points if they are identified as matching an entry within the dictionary. In other embodiments, the pseudo-singleton creates the end points in response to a determination that the end points do not match an entry within the dictionary. In some embodiments, establishment of the line requires no identification of the state of the end points by checks or analysis on the asynchronous message.

[0086] The preceding depiction of the collaborative CAx applications and other inventive elements described herein are intended to be illustrative rather than definitive. Similarly, the claimed 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 description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A method for multi-user CAx editing, the method comprising:

receiving a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment;

identifying a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element;

comparing the reference portion to a dictionary associated with the CAx environment;

receiving the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary; and

creating the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

2. The method of claim **1**, wherein the reference portion of the hierarchical message comprises parent data corresponding to the element.

3. The method of claim **1**, wherein the reference portion of the hierarchical message comprises child data corresponding to the element.

4. The method of claim **1**, wherein the hierarchical message comprises a neutral format.

5. The method of claim **1**, wherein comparing the reference portion to the dictionary comprises applying a dictionary evaluator having a pseudo-singleton pattern.

6. The method of claim **5**, wherein the pseudo-singleton pattern prevents duplicate instances of the element from being created.

7. The method of claim **5**, wherein the pseudo-singleton pattern operates a private constructor and applies a shared instance retrieval on the dictionary.

8. A computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions readable/executable by a processor to cause the processor to:

receive a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment;

identify a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element;

compare the reference portion to a dictionary of the CAx environment;

receive the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary; and

create the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

9. The computer program product of claim **8**, wherein the reference portion of the hierarchical message comprises parent data corresponding to the element.

10. The computer program product of claim **8**, wherein the reference portion of the hierarchical message comprises child data corresponding to the element.

11. The computer program product of claim **8**, wherein the hierarchical message comprises a neutral format.

12. The computer program product of claim **8**, wherein comparing the reference portion to the dictionary comprises applying a dictionary evaluator having a pseudo-singleton pattern.

13. The computer program product of claim **12**, wherein the pseudo-singleton pattern prevents duplicate instances of the element from being created.

14. The computer program product of claim **12**, wherein the pseudo-singleton pattern operates a private constructor and applies a shared instance retrieval on the dictionary.

15. A computer system comprising:

a processor;

a memory; and

a computer readable medium having instructions encoded thereon to cause the processor to:

receive a hierarchical message in an asynchronous client-server application, the hierarchical message corresponding to an element of a CAx model in a CAx environment;

identify a reference portion of the hierarchical message upon which a remaining portion of the element depends, the reference portion being unique to the element;

compare the reference portion to a dictionary of the CAx environment;

receive the reference portion of the element as a return output in response to identifying a match for the reference portion in the dictionary; and

create the reference portion of the element within the CAx environment in response to identifying no match for the reference portion in the dictionary.

16. The computer system of claim **15**, wherein the reference portion of the hierarchical message comprises parent data corresponding to the element.

17. The computer system of claim **15**, wherein the reference portion of the hierarchical message comprises child data corresponding to the element.

18. The computer system of claim **15**, wherein the hierarchical message comprises a neutral format.

19. The computer system of claim **15**, wherein comparing the reference portion to the dictionary comprises applying a dictionary evaluator having a pseudo-singleton pattern.

20. The computer system of claim **19**, wherein the pseudo-singleton pattern prevents duplicate instances of the element from being created.

* * * * *