US 20110179090A1

(19) **United States**

(12) **Patent Application Publication**

Sivaram et al.

(10) **Pub. No.: US 2011/0179090 A1**

(43) **Pub. Date: Jul. 21, 2011**

(54) **PRODUCT LIFECYCLE MANAGEMENT USING A SPARSELY POPULATED TABLE**

(75) Inventors: **Ashok Sivaram**, Troy, MI (US); **William A. Hinton**, Sterling Heights, MI (US); **Kaushal Patankar**, Pune (IN); **Sunil Viswanathan**, Cypress, CA (US)

(73) Assignee: **SIEMENS PRODUCT LIFECYCLE MANAGEMENT SOFTWARE INC.**, Plano, TX (US)

(21) Appl. No.: **13/009,944**

(22) Filed: **Jan. 20, 2011**

Related U.S. Application Data

(60) Provisional application No. 61/297,165, filed on Jan. 21, 2010, provisional application No. 61/297,158, filed on Jan. 21, 2010.

Publication Classification

(51) **Int. Cl.**

G06F 7/00	(2006.01)
G06F 17/30	(2006.01)

(52) **U.S. Cl.** **707/803**; 707/E17.044

(57) **ABSTRACT**

A product lifecycle management (PLM) system, method, and machine-readable medium. A method includes receiving a command to create a parameter definition for a PLM object. The method includes, in response to the command, creating a table definition object and a table object based on the table definition object. The method includes creating at least one cell associated with the table object of an object type that describes an object parameter, and using the table object as an attribute of the parameter definition.

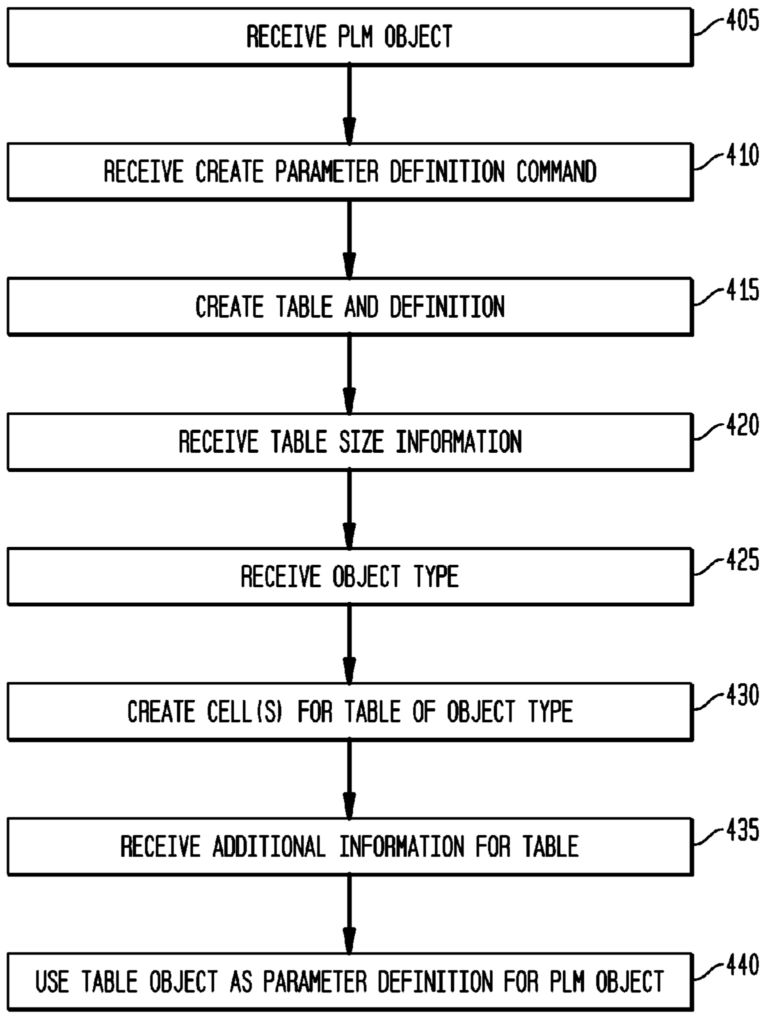

RECEIVE PLM OBJECT
405
RECEIVE CREATE PARAMETER DEFINITION COMMAND
410
CREATE TABLE AND DEFINITION
415
RECEIVE TABLE SIZE INFORMATION
420
RECEIVE OBJECT TYPE
425
CREATE CELL(S) FOR TABLE OF OBJECT TYPE
430
RECEIVE ADDITIONAL INFORMATION FOR TABLE
435
USE TABLE OBJECT AS PARAMETER DEFINITION FOR PLM OBJECT
440

FIG. 1

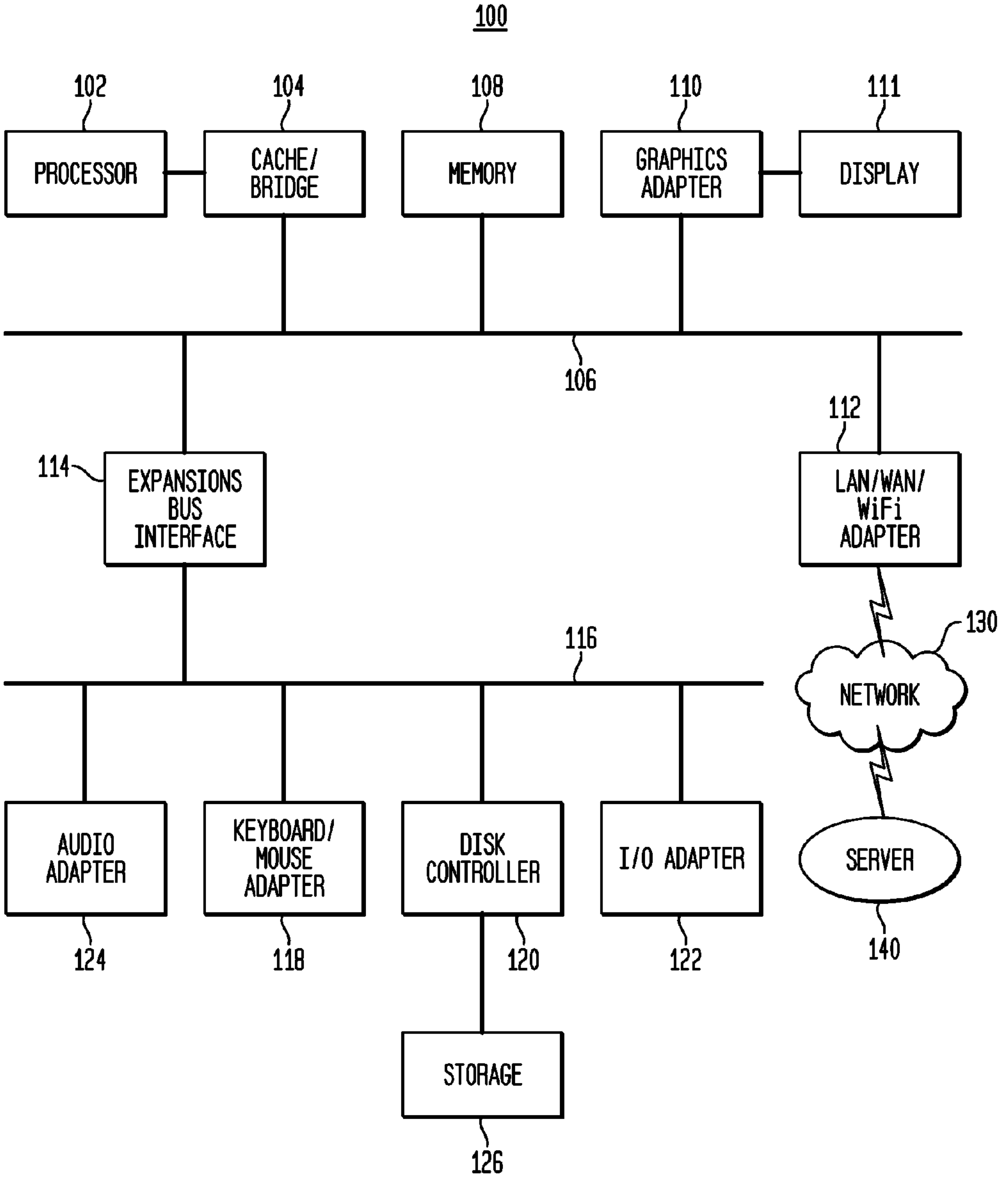

100
102
PROCESSOR
104
CACHE/ BRIDGE
108
MEMORY
110
GRAPHICS ADAPTER
111
DISPLAY
106
112
LAN/WAN/ WiFi ADAPTER
114
EXPANSIONS BUS INTERFACE
116
130
NETWORK
SERVER
140
AUDIO ADAPTER
124
KEYBOARD/ MOUSE ADAPTER
118
DISK CONTROLLER
120
I/O ADAPTER
122
STORAGE
126

FIG. 2

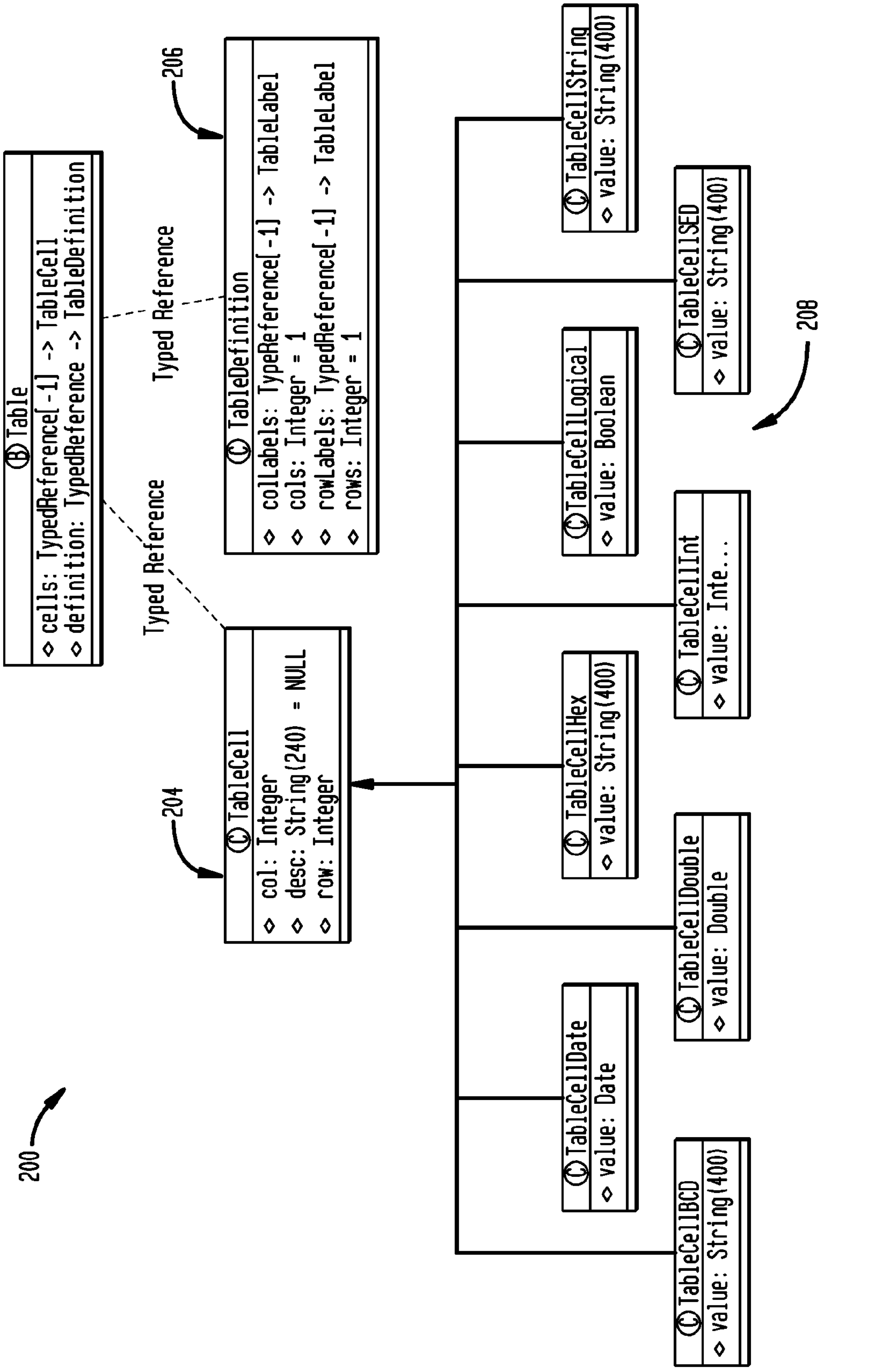
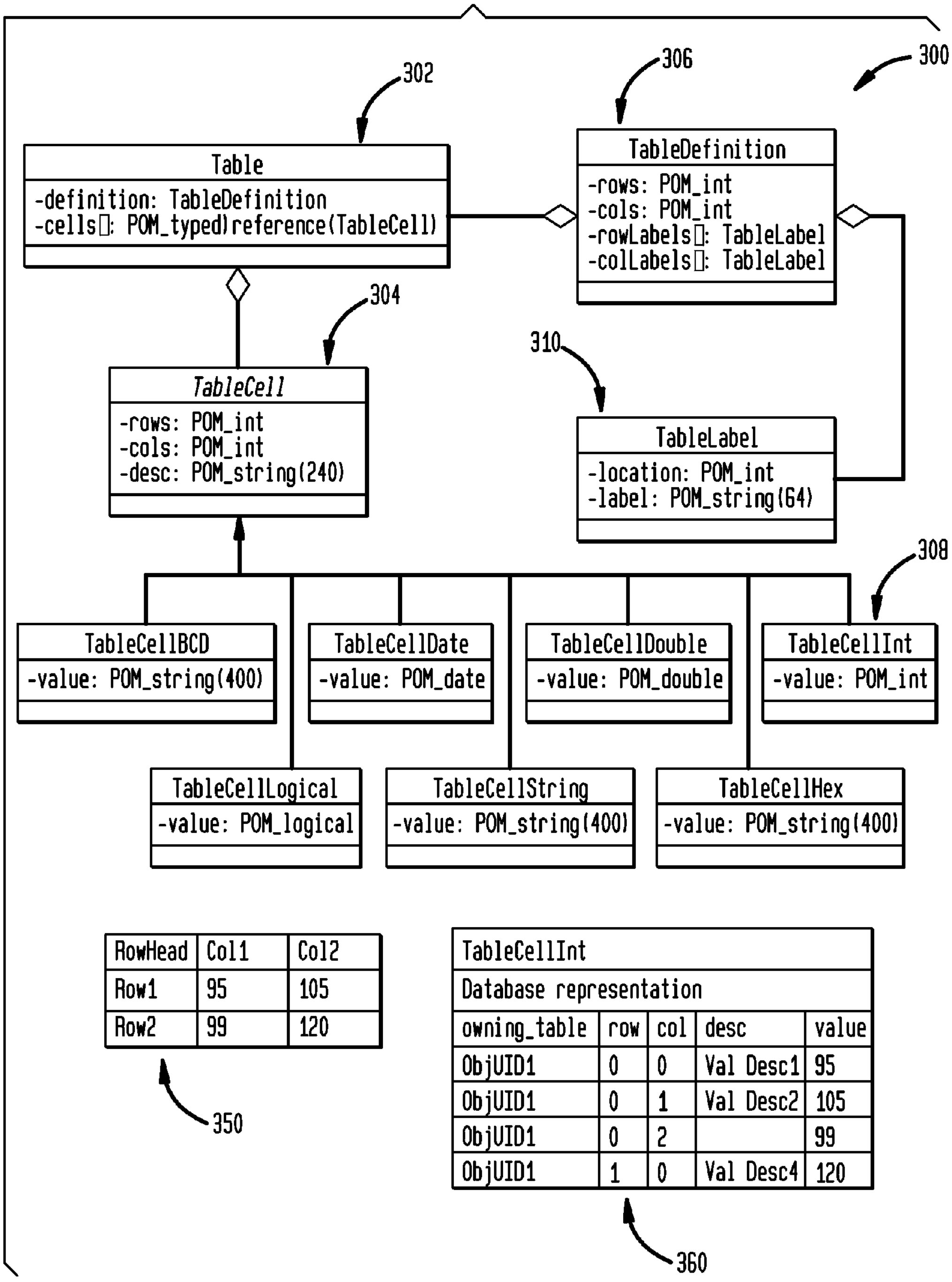

200
202
B Table
◇ cells: TypedReference[-1] -> TableCell
◇ definition: TypedReference -> TableDefinition
Typed Reference
Typed Reference
206
C TableDefinition
◇ colLabels: TypeReference[-1] -> TableLabel
◇ cols: Integer = 1
◇ rowLabels: TypedReference[-1] -> TableLabel
◇ rows: Integer = 1
204
C TableCell
◇ col: Integer
◇ desc: String(240) = NULL
◇ row: Integer
C TableCellString
◇ value: String(400)
C TableCellSED
◇ value: String(400)
C TableCellLogical
◇ value: Boolean
C TableCellInt
◇ value: Inte...
C TableCellHex
◇ value: String(400)
C TableCellDouble
◇ value: Double
C TableCellDate
◇ value: Date
C TableCellBCD
◇ value: String(400)
208

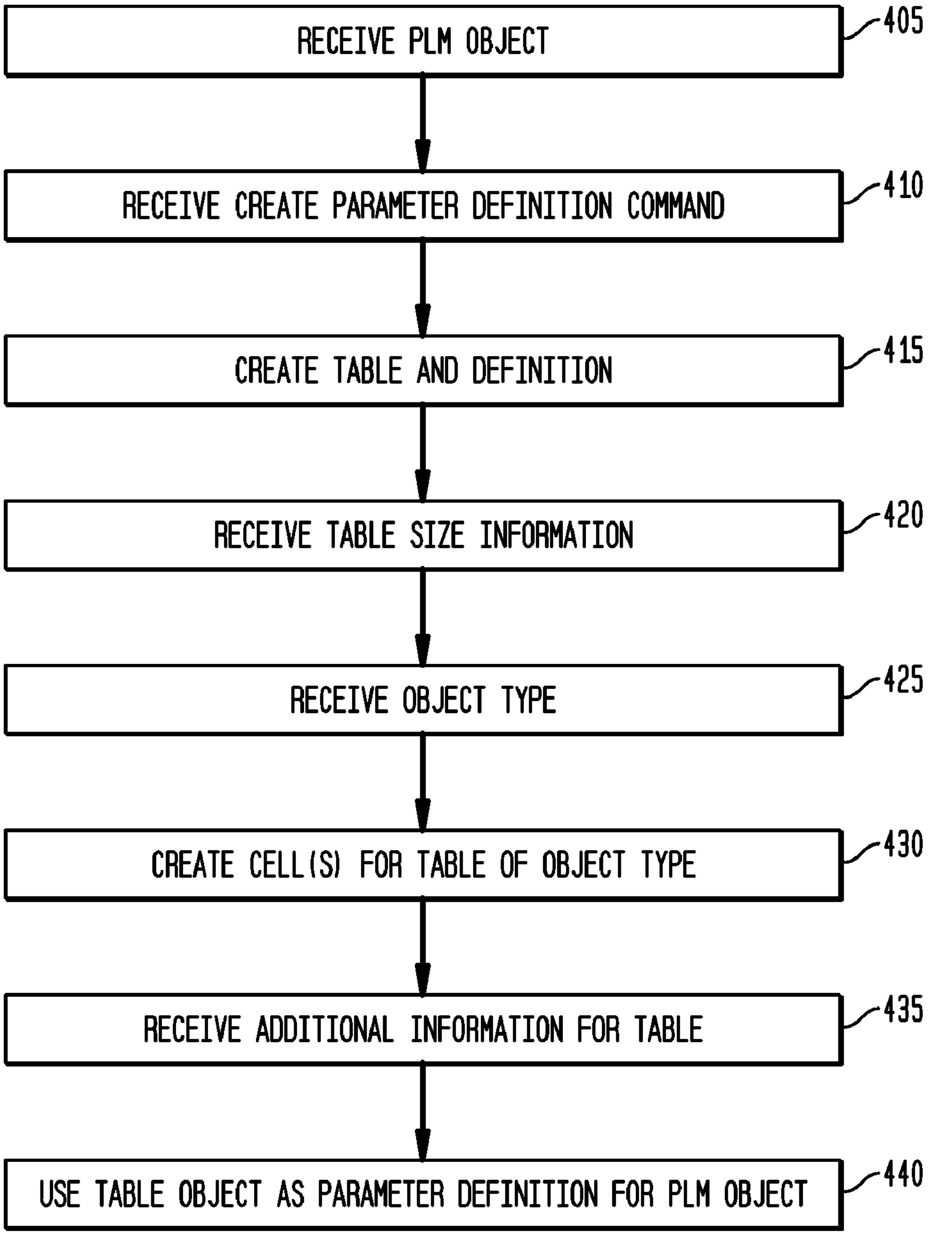
FIG. 3



RowHead	Col1	Col2
Row1	95	105
Row2	99	120

TableCellInt				
Database representation				
owning_table	row	col	desc	value
ObjUID1	0	0	Val Desc1	95
ObjUID1	0	1	Val Desc2	105
ObjUID1	0	2		99
ObjUID1	1	0	Val Desc4	120

FIG. 4


RECEIVE PLM OBJECT
405
RECEIVE CREATE PARAMETER DEFINITION COMMAND
410
CREATE TABLE AND DEFINITION
415
RECEIVE TABLE SIZE INFORMATION
420
RECEIVE OBJECT TYPE
425
CREATE CELL(S) FOR TABLE OF OBJECT TYPE
430
RECEIVE ADDITIONAL INFORMATION FOR TABLE
435
USE TABLE OBJECT AS PARAMETER DEFINITION FOR PLM OBJECT
440

PRODUCT LIFECYCLE MANAGEMENT USING A SPARSELY POPULATED TABLE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of the filing date of U.S. Provisional Patent Applications 61/297,165 and 61/297,158, both filed Jan. 21, 2010, which are hereby incorporated by reference. This application also includes some subject matter in common with concurrently-filed U.S. patent application Ser. No. ______ for "Adaptive Table Sizing for Multiple-Attribute Parameters", which is hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present disclosure is directed, in general, to systems and methods for computer-aided design, manufacturing, engineering, modeling, and visualization (individually and collectively, "CAD" and "CAD systems"), and to systems that manage product lifecycle data for manufacturers, companies, suppliers, and customers ("PLM systems").

BACKGROUND OF THE DISCLOSURE

[0003] Many manufactured products are first designed and modeled in CAD systems, and complex data is often managed in PLM systems. Improved systems are desirable.

SUMMARY OF THE DISCLOSURE

[0004] Various embodiments include systems, methods, and computer program products. Various embodiments can model table-type data with different cell types such as Integer, String, Double, Hexadecimal, Date, etc. in a PLM system. In various embodiments, these tables can be modeled with homogeneous (i.e., same type) cells or with cells of heterogeneous types and can be completely populated or sparsely populated tables.

[0005] A method includes receiving a command to create a parameter definition for a PLM object. The method includes, in response to the command, creating a table definition object and a table object based on the table definition object. The method includes creating at least one cell associated with the table object of an object type that describes an object parameter, and using the table object as an attribute of the parameter definition.

[0006] The foregoing has outlined rather broadly the features and technical advantages of the present disclosure so that those skilled in the art may better understand the detailed description that follows. Additional features and advantages of the disclosure will be described hereinafter that form the subject of the claims. Those skilled in the art will appreciate that they may readily use the conception and the specific embodiment disclosed as a basis for modifying or designing other structures for carrying out the same purposes of the present disclosure, Those skilled in the art will also realize that such equivalent constructions do not depart from the spirit and scope of the disclosure in its broadest form.

[0007] Before undertaking the DETAILED DESCRIPTION below, it may be advantageous to set forth definitions of certain words or phrases used throughout this patent document: the terms "include" and "comprise," as well as derivatives thereof, mean inclusion without limitation; the term "or" is inclusive, meaning and/or; the phrases "associated with" and "associated therewith," as well as derivatives thereof, may mean to include, be included within, interconnect with, contain, be contained within, connect to or with, couple to or with, be communicable with, cooperate with, interleave, juxtapose, be proximate to, be bound to or with, have, have a property of, or the like; and the term "controller" means any device, system or part thereof that controls at least one operation, whether such a device is implemented in hardware, firmware, software or some combination of at least two of the same. It should be noted that the functionality associated with any particular controller may be centralized or distributed, whether locally or remotely. Definitions for certain words and phrases are provided throughout this patent document, and those of ordinary skill in the art will understand that such definitions apply in many, if not most, instances to prior as well as future uses of such defined words and phrases. While some terms may include a wide variety of embodiments, the appended claims may expressly limit these terms to specific embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] For a more complete understanding of the present disclosure, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like numbers designate like objects, and in which:

[0009] FIG. **1** depicts a block diagram of a data processing system in which an embodiment can be implemented;

[0010] FIG. **2** shows a unified modeling language (UML) diagram of features in accordance with various embodiments;

[0011] FIG. **3** shows a UML diagram including additions to the foundation template that show the data model for a table cell in accordance with disclosed embodiments; and

[0012] FIG. **4** depicts a process in accordance with disclosed embodiments.

DETAILED DESCRIPTION

[0013] FIGS. **1** through **4**, discussed below, and the various embodiments used to describe the principles of the present disclosure in this patent document are by way of illustration only and should not be construed in any way to limit the scope of the disclosure. Those skilled in the art will understand that the principles of the present disclosure may be implemented in any suitably arranged device. The numerous innovative teachings of the present application will be described with reference to exemplary non-limiting embodiments.

[0014] Currently, there is no process for managing or modeling a sparsely populated table and representing these tables as attributes of an object that is managed within a PLM system. Current PLM systems cannot capture sparsely populated homogeneous or heterogeneous Table type data in such a way that it can be used as its own data model artifact.

[0015] Disclosed embodiments include a table definition or a table that can be reused on a completely different parameter definition if required. For example, applications in the calibration parameter management domain can require manipulation of Table type data, and in the absence of this construct in the PLM domain, parameter management through PLM is at best incomplete when it comes to managing complex parameters. Several other application domains which require manipulation of matrices and tables will also benefit from a Table representation as disclosed herein.

[0016] Various embodiments include systems and methods for modeling and managing table type data with different cell types such as Integer, String, Double, Hexadecimal, Date, etc, by a PLM system. These tables can be modeled with homogeneous cells or with cells of heterogeneous types and can be completely populated or sparsely populated tables.

[0017] FIG. **1** depicts a block diagram of a data processing system in which an embodiment can be implemented. The data processing system depicted includes a processor **102** connected to a level two cache/bridge **104**, which is connected in turn to a local system bus **106**. Local system bus **106** may be, for example, a peripheral component interconnect (PCI) architecture bus. Also connected to local system bus in the depicted example are a main memory **108** and a graphics adapter **110**. The graphics adapter **110** may be connected to display **111**.

[0018] Other peripherals, such as local area network (LAN)/Wide Area Network/Wireless (e.g. WiFi) adapter **112**, may also be connected to local system bus **106**. Expansion bus interface **114** connects local system bus **106** to input/output (I/O) bus **116**. I/O bus **116** is connected to keyboard/mouse adapter **118**, disk controller **120**, and I/O adapter **122**. Disk controller **120** can be connected to a storage **126**, which can be any suitable machine usable or machine readable storage medium, including but not limited to nonvolatile, hard-coded type mediums such as read only memories (ROMs) or erasable, electrically programmable read only memories (EEPROMs), magnetic tape storage, and user-recordable type mediums such as floppy disks, hard disk drives and compact disk read only memories (CD-ROMs) or digital versatile disks (DVDs), and other known optical, electrical, or magnetic storage devices.

[0019] Also connected to I/O bus **116** in the example shown is audio adapter **124**, to which speakers (not shown) may be connected for playing sounds. Keyboard/mouse adapter **118** provides a connection for a pointing device (not shown), such as a mouse, trackball, trackpointer, etc.

[0020] Those of ordinary skill in the art will appreciate that the hardware depicted in FIG. **1** may vary for particular implementations. For example, other peripheral devices, such as an optical disk drive and the like, also may be used in addition or in place of the hardware depicted. The depicted example is provided for the purpose of explanation only and is not meant to imply architectural limitations with respect to the present disclosure.

[0021] A data processing system in accordance with an embodiment of the present disclosure includes an operating system employing a graphical user interface. The operating system permits multiple display windows to be presented in the graphical user interface simultaneously, with each display window providing an interface to a different application or to a different instance of the same application. A cursor in the graphical user interface may be manipulated by a user through the pointing device. The position of the cursor may be changed and/or an event, such as clicking a mouse button, generated to actuate a desired response.

[0022] One of various commercial operating systems, such as a version of Microsoft Windows™, a product of Microsoft Corporation located in Redmond, Wash. may be employed if suitably modified. The operating system is modified or created in accordance with the present disclosure as described.

[0023] LAN/WAN/Wireless adapter **112** can be connected to a network **130** (not a part of data processing system **100**), which can be any public or private data processing system network or combination of networks, as known to those of skill in the art, including the Internet. Data processing system **100** can communicate over network **130** with server system **140**, which is also not part of data processing system **100**, but can be implemented, for example, as a separate data processing system **100**.

[0024] Several industries, from automotive, aircraft, satellite, and trains to ATM stations and parking meters, employ software-based calibration and configuration and on a regular basis deal with Table attributes for their parameters. Disclosed embodiments include a mechanism to capture table information as attributes on the main managed object. This table information can be sparsely populated so that some or all table values can be empty and populated later, and can include strongly typing each cell of the table. Without features as disclosed herein, there is no straightforward way to capture these types of attributes as part of the metadata of calibration and configuration parameters.

[0025] Various embodiments allow tables to be represented in a PLM data model just as one would represent a simple int, double, string or date data type, and further allow for strongly typing each element or cell of the table. FIG. **2** illustrates a unified modeling language (UML) example **200** of a table in accordance with disclosed embodiments. The embodiments described herein can be used with sparsely-populated tables and up to fully-populated tables, but provide significant advantages in reduced storage requirements for sparsely-populated tables.

[0026] According to various embodiments, the system represents the Table **202** as a basic class that contains a variable length array (VLA) of high level cell objects TableCell **204**. Table **202** contains a plurality of cells TableCell **204** and is defined by a TableDefinition **206**, which stores label and size information for the table **202**. Each TableCell **204** defines a type of cell for the table, and can have corresponding individual TableCell classes **208**, such as TableCellBCD, TableCellDate, TableCellDouble, TableCellInt, TableCellLogical, TableCellSED, and TableCellString.

[0027] The high level Cell class TableCell **204** has several specializations corresponding to the type of each cell. The specialization classes **208** encapsulate the value of the cell and this is where the strong typing occurs. Each specialized cell class holds the strongly typed value as an attribute in its member; for example, an int type cell is represented by strongly typing its value member to int and so on.

[0028] This provides various capabilities and advantages. For example, at the table level, heterogeneity of types can be maintained as it contains a variable length array (VLA) of cells and no typing occurs here. Further, at the Cell level strong typing can be achieved by typing the value member in the corresponding cell specialization.

[0029] A calibration or configuration parameter in a PLM system can have several attributes to be modeled. These hold different parameter metadata such as minimum value, maximum value, valid value, or initial value, etc. These are attributes of the parameter definition which are defined by software engineers or control engineers. These attributes can then be used downstream by software developers and suppliers to develop embedded software that use these parameter definitions. Later, when parameter specialists set values for use in given projects, the attributes will conform to the design intent of the parameter definition as originally specified by the software engineers as part of the parameter definition.

[0030] The values for these parameters, in context of specific projects, can be specified by a parameter specialist or other user and hold the actual parameter values based on the provided parameter definitions. Parameter values can be of different data types for different parameters. Therefore the support for different data types along with validation, extensibility, and customization need to be handled at the database level. Various design considerations related to the disclosed data model are described herein.

[0031] The data model preferably supports creation of parameters having different data types for its value attributes. Primitive data types like integer, double, string, logical, etc. and other data types like complex numbers with a single value, simple arrays and $(m \times n)^2$ dimensional arrays. In addition to using the primitive data types, the following types like Hex, Binary coded decimal, State Encoded, Bitmap can also to be supported

[0032] Various embodiments introduce generic classes that can be leveraged by other system to model Table Cells as described herein. Various embodiments introduce classes that create the various parameter definition, parameter grouping, and parameter grouping value objects.

[0033] The data model can utilize a template updated for Table representation Data model, the ability to manage one/two dimensional table of primitive types, the ability to manage row, column and value description, and the ability to provide and use a reusable table definition.

[0034] FIG. **3** shows an exemplary UML, diagram **300** including additions to the foundation template show the data model for a table cell in accordance with disclosed embodiments, along with a corresponding table **350** and flattened database representation **360** of this data, Table Table **302** is a basic class that contains a variable length array (VLA) of high level cell objects TableCell **304**. Table **302** contains cells **304** and is defined by a table definition TableDefinition **306**, which stores label and size information for the table **302**. Each call **304** defines a type of cell for the table, and can have corresponding individual TableCell classes **308**. In this example, a table label TableLabel **310** is associated with the table definition, and stores a label for the table.

[0035] In the example of FIG. **3**, RowHead, Col1, Col2, Row1, Row2 are the Table Headers, Column Headers and Row Headers, respectively, as shown in the table **350**. The values corresponding to the cells are 95, 105, and 98, 120. This is an example of a 2d-Integer table. As the attributes of the parameter definition like the minimum and maximum values etc also need to be arrays if the data type for the parameter definition is an array. In this example, table header, column headers, and row headers can be stored, for example, as cells **304** as a specific cell class **308** such as TableCellString, while the stored values can be, for example, as a cell class **308** such as TableCellInt.

[0036] The flattened representation **360** of this table shows how each cell of the table would be stored. Essentially, each row in the flattened representation is a cell of the table. The Owning Table field would reflect the same unique identifier (UID) as Table table **302**.

[0037] Note that, by this mechanism one can also attach descriptions to individual cells by specifying a description in a "desc" attribute.

[0038] The row and column headers are not stored in the cell classes because they need to be stored at the table level and not at the cell level. The row and column headers are defined in the TableDefinition class and their labels are stored in the TableLabel class.

[0039] FIG. **4** depicts a flowchart of a process in accordance with disclosed embodiments.

[0040] A PLM system receives a PLM object (step **405**). This can be, for example, a part or component for an assembly, or otherwise. The PLM system can be implemented, for example, as a data processing system **100**. "Receiving", as used herein, can include loading from storage, receiving from another process or data processing system, receiving through an interaction with a user, or otherwise.

[0041] The PLM system receives a command to create a parameter definition for the PLM object (step **410**).

[0042] In response the system creates an initial parameter definition for the PLM object, as described herein, a table definition object associated with the parameter definition, and a table object for the parameter definition based on the table definition object (step **415**). Of course, if any of these already exist in the system for the object, the system can re-use and modify it instead of re-creating it. The table object can be stored in or as associated with the parameter definition for the PLM object, or otherwise.

[0043] The system receives label and size information for the table, and stores it in the table definition object (step **420**). This step can include prompting a user for this information.

[0044] The system receives an object type for the table (step **425**). This type can be, for example, Integer, String, double precision (Double), Boolean, Date, stream editor (SED), Hex, binary-coded decimal (BCD). BitMap, or other data types known to those of skill in the art.

[0045] The system creates at least one cell, associated with the table object, of the received object type (step **430**). This step can include prompting a user for this information. The object type specifies the type of object used to describe the object parameter. The at least one cell can be a contained in a variable length array associated with or stored in the table object.

[0046] The system can receive additional table information for the table (step **435**). This additional table information can describe the table or the parameters for the PLM object, and can include, for example, a name, description, unique identifier corresponding to the PLM object, Size Units, Size, Resolution, Rows, Columns, row descriptors, column descriptors, or other information. This additional table information can be stored in the at least one cell or in the table definition, depending on the type of information.

[0047] The system thereafter uses the table object, including its cells and table definition, as an attribute of the parameter definition for the PLM object (step **440**). The table object can be stored as a sparsely-populated table. This step can include displaying the PLM object, with or without an associated assembly, in accordance with the parameter definition.

[0048] In some embodiments, a method includes receiving a command to create a parameter definition for a PLM object. The method includes, in response to the command, creating an initial parameter definition, creating a table definition object, if one did not already exist in the system, and creating a table object based on that definition. If a suitable table definition did exist, the method includes reusing the existing table definition on a newly created table. The method includes creating at least one cell associated with the table object of an object type that describes an object parameter, and using the table object in the parameter definition as an attribute of that parameter definition.

[0049] In various embodiments, various steps described herein can be omitted, repeated, performed sequentially or concurrently, performed in a different order, or otherwise, as described by the claims below. For example, in many cases, multiple cells will be created for the table object, and the system will receive and store information in each of the created cells.

[0050] Those skilled in the art will recognize that, for simplicity and clarity, the full structure and operation of all data processing systems suitable for use with the present disclosure is not being depicted or described herein. Instead, only so much of a data processing system as is unique to the present disclosure or necessary for an understanding of the present disclosure is depicted and described. The remainder of the construction and operation of data processing system **100** may conform to any of the various current implementations and practices known in the art. While specific embodiments are described in terms of PLM systems and data, other embodiments can include other CAD system and data, or other data processing systems and data, that perform processes as described herein.

[0051] It is important to note that while the disclosure includes a description in the context of a fully functional system, those skilled in the art will appreciate that at least portions of the mechanism of the present disclosure are capable of being distributed in the form of a instructions contained within a machine-usable, computer-usable, or computer-readable medium in any of a variety of forms, and that the present disclosure applies equally regardless of the particular type of instruction or signal bearing medium or storage medium utilized to actually carry out the distribution. Examples of machine usable/readable or computer usable/readable mediums include: nonvolatile, hard-coded type mediums such as read only memories (ROMs) or erasable, electrically programmable read only memories (EEPROMs), and user-recordable type mediums such as floppy disks, hard disk drives and compact disk read only memories (CD-ROMs) or digital versatile disks (DVDs).

[0052] Although an exemplary embodiment of the present disclosure has been described in detail, those skilled in the art will understand that various changes, substitutions, variations, and improvements disclosed herein may be made without departing from the spirit and scope of the disclosure in its broadest form.

[0053] None of the description in the present application should be read as implying that any particular element, step, or function is an essential element which must be included in the claim scope: the scope of patented subject matter is defined only by the allowed claims. Moreover, none of these claims are intended to invoke paragraph six of 35 USC §112 unless the exact words "means for" are followed by a participle.

What is claimed is:

1. A method, performed by a product lifecycle management (PLM) system, comprising:

receiving a command to create a parameter definition for a PLM object;

in response to the command, creating a table definition object and a table object based on the table definition object;

creating at least one cell associated with the table object of an object type that describes an object parameter;

using the table object as an attribute of the parameter definition.

2. The method of claim **1**, wherein the PLM system receives label and size information for the table object and stores the label and size information in the table definition object.

3. The method of claim **1**, wherein the PLM system receives the object type.

4. The method of claim **1**, wherein the object type is one of Integer, String, Double, Boolean, Date, SED, Hex, BCD, or BitMap.

5. The method of claim **1**, wherein the at least one cell is contained in a variable length array associated with the table object.

6. The method of claim **1**, wherein the PLM system receives additional table information for the table that includes at least one of a name, a description, unique identifier corresponding to the PLM object, size units, a size, a resolution, row descriptors, and column descriptors.

7. The method of claim **1**, wherein the table object is stored as a sparsely-populated table.

8. A product lifecycle management (PLM) system, comprising:

a processor; and

a storage connected to be accessed by the processor, wherein the PLM system is configured to perform the steps of

receiving a command to create a parameter definition for a PLM object;

in response to the command, creating a table definition object and a table object based on the table definition object;

creating at least one cell associated with the table object of an object type that describes an object parameter;

using the table object as an attribute of the parameter definition.

9. The PLM system of claim **8**, wherein the PLM system receives label and size information for the table object and stores the label and size information in the table definition object.

10. The PLM system of claim **8**, wherein the PLM system receives the object type.

11. The PLM system of claim **8**, wherein the object type is one of Integer, String, Double, Boolean, Date, SED, Hex, BCD, or BitMap.

12. The PLM system of claim **8**, wherein the at least one cell is contained in a variable length array associated with the table object.

13. The PLM system of claim **8**, wherein the PLM system receives additional table information for the table that includes at least one of a name, a description, a unique identifier corresponding to the PLM object, size units, a size, a resolution, row descriptors, and column descriptors.

14. The PLM system of claim **8**, wherein the table object is stored as a sparsely-populated table.

15. A tangible machine-readable medium encoded with executable instructions, that, when executed, cause a product lifecycle management (PLM) system to perform the steps of:

receiving a command to create a parameter definition for a PLM object;

in response to the command, creating a table definition object and a table object based on the table definition object;

creating at least one cell associated with the table object of an object type that describes an object parameter;

using the table object as an attribute of the parameter definition.

16. The machine-readable medium of claim **15**, wherein the instructions further cause the PLM system to receive label and size information for the table object and stores the label and size information in the table definition object.

17. The machine-readable medium of claim **15**, wherein the instructions further cause the PLM system to receive the object type.

18. The machine-readable medium of claim **15**, wherein the object type is one of Integer, String, Double, Boolean, Date, SED, Hex, BCD, or BitMap.

19. The machine-readable medium of claim **15**, wherein the at least one cell is contained in a variable length array associated with the table object.

20. The machine-readable medium of claim **15**, wherein the instructions further cause the PLM system to receive additional table information for the table that includes at least one of a name, a description, a unique identifier corresponding to the PLM object, size units, a size, a resolution, row descriptors, and column descriptors.

21. The machine-readable medium of claim **15**, wherein the table object is stored as a sparsely-populated table.

* * * * *