(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 November 2012 (01.11.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/148681 A1

(51) International Patent Classification:
G06F 9/48 (2006.01) *G06F 21/00* (2006.01)
G06Q 10/00 (2012.01)

(21) International Application Number:
PCT/US2012/033180

(22) International Filing Date:
12 April 2012 (12.04.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/480,527 29 April 2011 (29.04.2011) US
13/240,463 22 September 2011 (22.09.2011) US

(71) Applicant *(for all designated States except US)*: **SIEMENS PRODUCT LIFECYCLE MANAGEMENT SOFTWARE INC.** [US/US]; 5800 Granite Parkway, Suite 600, Plano, Texas 75024 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **INSKO, Matthew J.** [US/US]; 28 McCormick Trail, Milford, Ohio 45150 (US). **IYER, Niranjan K.** [IN/US]; 3525A Del Mar Heights Rd., #116, San Diego, California 92130 (US).

(74) Agents: **WALLACE, JR., Michael J.** et al.; Siemens Corporation- Intellectual Property Dept., 170 Wood Avenue South, Iselin, New Jersey 08830 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: CROSS-SCHEDULE DEPENDENCIES USING PROXY TASKS

(57) Abstract: Product Data Management systems (100) require a way to create dependencies (240, 250) between tasks (212, 214,..) that reside in various schedules (210, 220) or between different schedules (210, 220). Disclosed embodiments comprise proxy tasks (218) for creation of cross schedule dependencies (250) without regard to the need of a master schedule or the need for both the schedules 210, 220) to be loaded simultaneously during creation A proxy task (218) corresponds to and is a substantial copy of the first task. With the disclosed embodiments security access is maintained while opening or working with a schedule that has a dependency (260) to another.

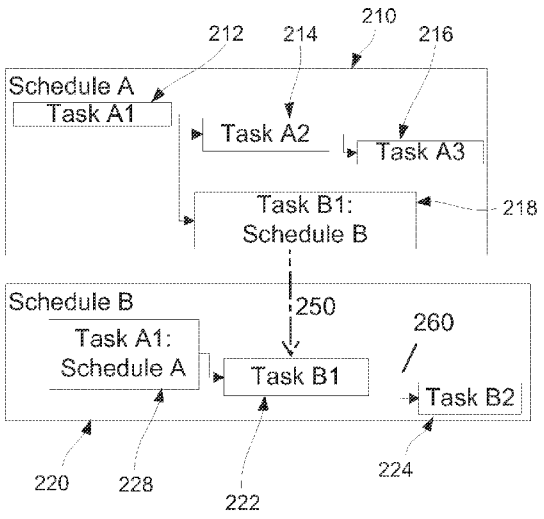

FIG. 2B

WO 2012/148681 A1

Published:

— *with international search report (Art. 21(3))*

Cross-Schedule Dependencies Using Proxy Tasks

[0001] This invention relates to a method, a system and a computer readable medium, for cross-schedule dependencies using proxy tasks according to the independent claims.

TECHNICAL FIELD

[0002] The present disclosure is directed, in general, to computer-aided design, visualization, and manufacturing systems ("CAD" systems), product lifecycle management ("PLM") systems, program and project management systems, and systems that manage data for products and other items (individually and collectively, product data management ("PDM") systems).

BACKGROUND

[0003] PDM systems can aid users in creating, managing, and executing project schedules, among other functions, including the scheduling of tasks.

SUMMARY OF THE DISCLOSURE

[0004] Various disclosed embodiments relate to systems and methods for managing cross-schedule dependencies using proxy tasks, and in particular, in PDM systems configured to perform processes as described herein.

[0005] Various embodiments include PDM systems, methods, and mediums. A method includes receiving a first schedule. The first schedule includes at least a first task. The method includes creating a proxy task in a second schedule. The proxy task corresponds to and is a substantial copy of the first task. The method includes storing the second schedule.

[0006] The method can include creating a dependency between a second task in the second schedule and the proxy task in the second schedule.

[0007] 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.

[0008] 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

[0009] 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:

[0010] Figure 1 depicts a block diagram of a data processing system in which an embodiment can be implemented;

[0011] Figures 2A and 2B illustrate proxy tasks and cross-schedule dependencies behavior in accordance with disclosed embodiments; and

[0012] Figures 3 and 4 depict flowcharts of processes in accordance with disclosed embodiments.

DETAILED DESCRIPTION

[0013] Figures 1 through 4, the other figures herein and attached hereto, 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] PDM systems and other systems require a way to create dependencies between tasks that reside in various schedules or between different schedules. This can entail requiring a master schedule that brings in both the schedules together as a "master schedule" and a "sub-schedule" to enable creating dependencies. Such cases also require both the schedules to be loaded at the same time to maintain data integrity. It is also very important that proper security access rules are adhered to while opening a schedule that contains cross schedule dependency. For example, if a Schedule A has a cross-schedule dependency to a Schedule B, and a user who has access only to Schedule A opens Schedule A, the user should be able to determine that there is a cross schedule dependency. However, the user cannot access or view any data in Schedule B.

[0015] Working with multiple schedules and having the need to create and work with cross-schedule dependencies can be an issue with customers using PDM systems. Requiring a master schedule as a pre-requisite or needing to load both the schedules to create a cross schedule dependency is too restrictive.

[0016] Disclosed embodiments include systems and methods for using proxy tasks for creation of cross schedule dependencies without regard to the need of a master schedule or the need for both the schedules to be loaded simultaneously during creation. Various embodiments disclosed herein support these operations in a source-independent manner, working equally well on schedules both independent and part of a master schedule.

[0017] Disclosed embodiments also maintain security access while opening or working with a schedule that has a dependency to another. For example, if a user cannot view the other schedule, then the system maintains that restriction. Similarly, if a user has only "view" access to the second schedule, but has "edit" access to the first schedule, then various embodiments disclosed herein can ensure that any changes they make in the first schedule do not modify the data of the second schedule. As such, systems and methods disclosed herein are well-suited to environments that require adherence to security.

[0018] Figure 1 depicts a block diagram of a data processing system in which an embodiment can be implemented, including as a PDM system particularly configured to perform processes as described herein. 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.

[0019] 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.

[0020] 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.

[0021] Those of ordinary skill in the art will appreciate that the hardware depicted in Figure 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.

[0022] 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.

[0023] 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.

[0024] 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.

[0025] Various embodiments described herein include systems and methods that can use "proxy tasks" to create and represent cross schedule dependencies. These proxy tasks

can be used to create cross schedule dependencies, and can also be used and placed anywhere within a schedule where it is meaningful.

[0026] As used herein, a "proxy task" is a representation of a task in a different schedule. It is used as a reference to a task outside the current schedule. It mirrors some or all of the attributes of the task or tasks represented in the other schedule to which the user has access. The proxy task can be used to represent a task, a milestone, a summary task or even an entire schedule. A proxy task enables users to mirror the task in another schedule and create cross schedule dependencies to it.

[0027] Various embodiments include specific characteristics of proxy tasks and their use in creation of cross-schedule dependencies. When a cross-schedule dependency is created between two tasks in different schedules, the system can create two proxy tasks, one corresponding to each schedule, and can also create the dependencies between the appropriate task and proxy tasks to indicate cross-schedule dependency.

[0028] Figures 2A and 2B illustrate proxy tasks and cross-schedule dependencies behavior in accordance with disclosed embodiments. These figures show a first schedule, Schedule A 210, and a second schedule, Schedule B 220.

[0029] Figure 2A shows that Schedule A 210 includes a Task A1 212, Task A2 214, and Task A3 216. Figure 2A also shows that Schedule B 220 includes a Task B1 222 and Task B2 224. The arrows indicate a dependency relationship between these tasks.

[0030] In the Fig. 2A, the bold arrow line indicates a cross-schedule dependency 240 between Task A1 212 and Task B1 222 in different schedules, Schedule A 210 and Schedule B 220. In a conventional system, such cross-schedule dependencies cause a number of problems as described above.

[0031] Figure 2B shows that Schedule A 210 includes a Task A1 212, Task A2 214, and Task A3 216. Figure 2B also shows that Schedule B 220 includes a Task B1 222 and Task B2 224. The arrows indicate a dependency relationship between these tasks.

[0032] Figure 2B illustrates that when a cross-schedule dependency is created, in accordance with disclosed embodiments, no direct cross-schedule dependency is represented as in Fig. 2A. Instead, the system creates two proxy tasks that reference the other schedule and task. In the example of Fig. 2B, ProxyTask B1 218 is created in Schedule A 210, that points to "Task B1: Schedule B". Similarly, ProxyTask A1 228 is created in Schedule B 220, that points to "Task A1: Schedule A". When these dependencies to proxy tasks are created, the system also stores an indication that a cross schedule dependency is created. Of course "ProxyTask" is an arbitrary name used in this example for purposes of illustration, and does not limit the claimed embodiments.

[0033] Disclosed embodiments create and maintain proxy tasks as separate objects and use them for cross-schedule dependency; this can aid in maintaining cohesiveness by keeping tasks that belong to a schedule in that schedule instead of referencing a task in another schedule.

[0034] Disclosed embodiments also enable greater support for security access in terms of dis-allowing a change in one schedule to be automatically propagated to another schedule if the user does not have the required permissions to do so. Program and project managers can better manage the schedules they own by not having their schedule plans change without their knowledge. Various embodiments also provide support for schedule validation rules to maintain data integrity. This mechanism leverages and supports the various kinds of dependencies that need to be supported in a program/project management system, such as Finish to Start (FS), Start to Start (SS), Finish to Finish (FF), Start to Finish (SF), etc.

[0035] Proxy tasks can also include external tasks or objects in a different system. This allows the opening up of the system to create cross schedule dependency to not just another schedule, but to external objects.

[0036] In various embodiments, the system can interact with a user in multiple ways to receive an input to create a proxy task. For example, the system can create a proxy task independently "from scratch" in response to a user pointing to the original task. The system can create a proxy task in response to a user using the system "clipboard" by

selecting a normal task and using a "paste as proxy" input on a different schedule to easily create a proxy task.

[0037] The system can also create proxy tasks as "mirrors" of tasks or items in a master schedule. For example, a user will be able to mirror a task in the master schedule as a proxy task in all the sub schedules, and the system creates corresponding proxy tasks. One benefit of this approach is that it enables program or project managers to very easily replicate and represent key milestones from a master schedule to all the participating sub schedules.

[0038] For example, when the system receives a user input to create a cross-schedule dependency from within a master schedule, the system can create two proxy tasks internally in each of the different schedules. For example, in response to a cross-schedule dependency from Task A in Schedule A to Task B in Schedule B in Fig. 2B, the system can create two proxy tasks – a proxy task (such as ProxyTask B 218) in Schedule A that represents the real task B and another proxy task (such as ProxyTask A 228) in Schedule B that represents the real task A. This enables the system to provide visibility of a dependent task in another schedule.

[0039] Systems and methods disclosed herein to use proxy tasks for cross-schedule dependencies enable users to create cross-schedule dependencies without the need to have master schedules. This benefits program and project managers in particular since they can define a project plan with dependencies to any schedules internal or external and thus not restricted to master schedules.

[0040] Further, disclosed techniques can create cross-schedule dependency without the need for both the schedules to be loaded at the same time thus benefiting any performance considerations.

[0041] Various embodiments also allow various schedule validations to be run and adhered to without the need to load both the schedules. For example, if a first-schedule cross-schedule dependency needs to be changed to a second schedule, it allows for that

without having the need for both the schedules to be loaded for validation thus benefiting performance reducing memory consumption.

[0042] Various embodiments provide different ways to create a cross schedule dependency, can create a cross-schedule dependency from a master schedule context, can create a dependency to or from a proxy task, can search for target tasks and create cross schedule dependencies, or can support different kinds of dependencies as described herein.

[0043] Proxy tasks can be created independently without dependencies as well. In those cases, it can mirror the attributes of the real task, and when the real task attributes change (e.g. the start dates, end dates, or any other attribute), the proxy task attributes can automatically reflect those changes. This can be important in cases where it is important for a plan to know the key drivers and milestones of a program being tracked in a master company plan.

[0044] The system can allow users to navigate from a proxy task to a real task, as well as relocate its position from one place to another in the structure without modifying the dates. This provides flexibility and easy navigability of proxy tasks.

[0045] Figure 3 depicts a process in accordance with disclosed embodiments that can be performed by one or more PLM systems. This process can be used to create a proxy task. In the processes described herein, the labels of "first schedule" and "second schedule" are arbitrary and used only to describe different schedules; these labels are not intended to be limiting.

[0046] The system receives a first schedule (step 305). "Receiving", as used herein, can include loading from storage, receiving from another device or process, receiving via an interaction with a user, or otherwise. The first schedule has at least one task.

[0047] The system searches the first schedule for any tasks for which to make a proxy (step 310). In some cases, this can be a search for a specific task in the first schedule that is intended to have a dependency relationship with or otherwise correspond to a not-yet-created proxy task in a second schedule.

[0048] The system selects the task in the first schedule (step 315). This selection can be performed automatically in response to identifying the task as a result of the search, or can include displaying the results of the search to a user and receiving a user selection of the task in the first schedule that is intended to have a dependency relationship with or otherwise correspond to a not-yet-created proxy task in the second schedule.

[0049] The system creates a proxy task in a second schedule, corresponding to the selected task in the first schedule (step 320). The proxy task can include information referencing the selected task in the first schedule, and the selected task in the first schedule can include information referencing the proxy task in the second schedule. The system can create and maintain dependency relationships between the proxy task and the selected task in the first schedule or in the second schedule. The proxy task can be a substantial duplicate of the selected task, meaning that it can include the same task data such as start, finish, work complete, constraint, status, etc.

[0050] The system stores the second schedule (step 325). As part of this step, the system can also store the first schedule.

[0051] Figure 4 depicts a process in accordance with disclosed embodiments that can be performed by one or more PLM systems. This process can be used to create a proxy task with dependencies.

[0052] The system receives a first schedule (step 405). The first schedule has at least one task. This schedule can be a master schedule.

[0053] The system receives at least one second schedule (step 410). In some cases, this can be one or more sub-schedules of the master schedule. For clarity of illustration, the second schedule will be referred to in the singular, but is not so limited.

[0054] The system creates a proxy task in the second schedule corresponding to the task in the first schedule (step 415). In some cases, this can be performed in response to receiving a user input, such as by the user "dragging" the task in the first schedule and "dropping" it into the second schedule. The proxy task can include information

referencing the task in the first schedule, and the task in the first schedule can include information referencing the proxy task.

[0055] The system can create and maintain dependency relationships between the proxy task and the task in the first schedule (step 420). As part of this step, the system can automatically detect any changes in the task in the first schedule and make corresponding changes to the proxy task. The system can also maintain dependencies between the proxy task (in the second schedule) and other tasks in the second schedule. In those cases, the system can perform a plurality of processes to detect any changes in the task in the first schedule, make corresponding changes to the proxy task in the second schedule, then also make corresponding changes to any other tasks in the second schedule that are dependent on the proxy task.

[0056] Note that in some cases, where there may be a "two way" dependency between the task in the first schedule and the tasks in the second schedule, where a task in the first schedule is dependent on a task in the second schedule, and a task in the second schedule is dependent on a task in the first schedule. In such cases, the system can create respective proxy tasks in both the first and second schedules, so that the dependency for each task in each schedule is to the proxy task in that same schedule, and the proxy tasks in each schedule can be updated according to changes in a corresponding task in the other schedule.

[0057] The system stores the second schedule (step 425). As part of this step, the system can also store the first schedule.

[0058] Note that creating a proxy task does not necessarily automatically create dependencies or require creation of dependencies. In certain embodiments, the system can automatically create proxy tasks in response to a user creating cross schedule dependencies.

[0059] Unless otherwise described, the various processes, actions, and steps described above can be performed concurrently, sequentially, in a different order, or omitted in

various embodiments. Various elements and steps described above can be combined differently or alternatively in other embodiments.

[0060] According to various embodiments, cross-schedule dependencies can be created between any schedules in the system using proxy tasks. Proxy tasks do not require a master schedule to create cross schedule dependencies, and does not require loading of both the schedules in its entirety to create cross schedule dependency. A proxy task can represent any type of task, milestone, phase, gate, schedule, or other event or object in the system, all referred to generically herein as "tasks". Proxy tasks can also be used to "mirror" tasks in sub-schedules from a master schedule. The system maintains security access rights of each task and their schedule including proxy tasks. The system can maintain corresponding drawing representations, for example, for a milestone proxy or a summary proxy.

[0061] In some embodiments, the system does not allow proxy tasks to be moved or edited directly, but instead they reflect the dates and attribute changes according to the "home" or original task. Proxy tasks can exist without a cross-schedule dependency existing.

[0062] Various embodiments include other PDM systems, methods, and mediums. For example, another method includes receiving a first schedule that includes at least a first task. The method includes creating a proxy task in a second schedule or in the same schedule. The proxy task corresponds to and is a substantial copy of the first task. The method includes creating a relation between the proxy task and the original task and creating a dependency between a proxy task and a normal schedule task. The method can include storing both the first and second schedule.

[0063] 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.

[0064] 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 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).

[0065] 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.

List of used reference numerals, glossary

100	data processing system, product data management data processing system; PDM data processing system
102	processor
104	cache/bridge
106	local system bus
108	main memory
110	graphics adapter
111	display
112	Local Area Network / Wide Area Network / Wireless adapter
114	expansion bus interface
116	input/output bus
118	keyboard/mouse adapter
120	disk controller
122	I/O adapter
124	audio adapter
130	network
140	server system
210	Schedule A
212	Task A1
214	Task A2
216	Task A3
218	ProxyTask B1

220 Schedule B

222 TaskB1

224 Task B2

228 ProxyTask A1

240 cross-schedule dependency between tasks

250 ProxyTask B1 218 pointing to Task B1: Schedule B

260 dependency relationship between tasks in a schedule

305 receiving a first schedule

310 searching the first schedule for any tasks, searching for a specific task in the first schedule

315 selecting the task in the first schedule

320 creating a proxy task in second schedule

325 storing the second schedule

405 receiving a first schedule

410 receiving at least one second schedule

415 creating a proxy task in the second schedule corresponding to the task in the first schedule

420 create dependency relationships between the proxy task and the task in the first schedule

425 storing the second schedule

CAD Computer Aided Design

PDM product data management

PLM product lifecycle management

Claims

1. A method performed by a product data management (PDM) data processing system (100), comprising:

- receiving (305) a first schedule (210), the first schedule (210) including at least a first task (212);
- creating (415) a proxy task (228) in a second schedule (220), the proxy task (228) corresponding to and being a copy of the first task (212);
- creating (420) a dependency (260) between a second task (222) in the second schedule (220) and the proxy task (228) in the second schedule (220); and
- storing (425) the second schedule (220).

2. The method of claim 1, wherein the first schedule (210) is a master schedule and the second schedule (220) is one of a plurality of sub-schedules.

3. The method of claim 1 or 2 , wherein the second schedule (220) is stored along with information referencing the first task.

4. The method of one of the claims 1 to 3, wherein the proxy task (218) is created in response to receiving a user dragging the first task (212) from the first schedule (210) and dropping it into the second schedule (220).

5. A product data management data processing system (100), comprising:
at least one processor (102); and
- an accessible memory, wherein the data processing system (100) is configured to
- receive a first schedule, the first schedule including at least a first task;
- create a proxy task in a second schedule, the proxy task corresponding to and being a substantial copy of the first task; and
- store the second schedule.

6. The data processing system (100) of claim 5, further configured to create a dependency between a second task in the second schedule and the proxy task in the second schedule.

7. The data processing system (100) of claim 5 or 6 , further configured to create a dependency between the proxy task and a second task in the second schedule.

8. The data processing system (100) of one of the claims 5 to 7, wherein the proxy task is created in response to receiving a user input.

9. The data processing system (100) of one of the claims 5 to 8, wherein the proxy task is created in response to receiving a user dragging the first task from the first schedule and dropping it into the second schedule.

10. The data processing system (100) of one of the claims 5 to 9, wherein the first schedule is a master schedule and the second schedule is one of a plurality of sub-schedules.

11. The data processing system (100) of one of the claims 5 to 10, wherein the second schedule is stored along with information referencing the first task.

12. A non-transitory computer-readable medium encoded with executable instructions that, when executed, cause a product data management data processing system (100) to:

- receive a first schedule, the first schedule including at least a first task;
- create a proxy task in a second schedule, the proxy task corresponding to and being a substantial copy of the first task; and
- store the second schedule.

13. The computer-readable medium of claim 12, wherein the instructions cause the data processing system (100) to create a dependency between a second task in the second schedule and the proxy task in the second schedule.

14. The computer-readable medium of claim 12 or 13, wherein the instructions cause the data processing system to create a dependency between the proxy task and a second task in the second schedule.

15. The computer-readable medium of one of the claims 12 to 14, wherein the proxy task is created in response to receiving a user input.

16. The computer-readable medium of one of the claims 12 to 15, wherein the proxy task is created in response to receiving a user dragging the first task from the first schedule and dropping it into the second schedule.

17. The computer-readable medium of one of the claims 12 to 16, wherein the first schedule is a master schedule and the second schedule is one of a plurality of sub-schedules.

18. The computer-readable medium of one of the claims 12 to 17, wherein the second schedule is stored along with information referencing the first task.

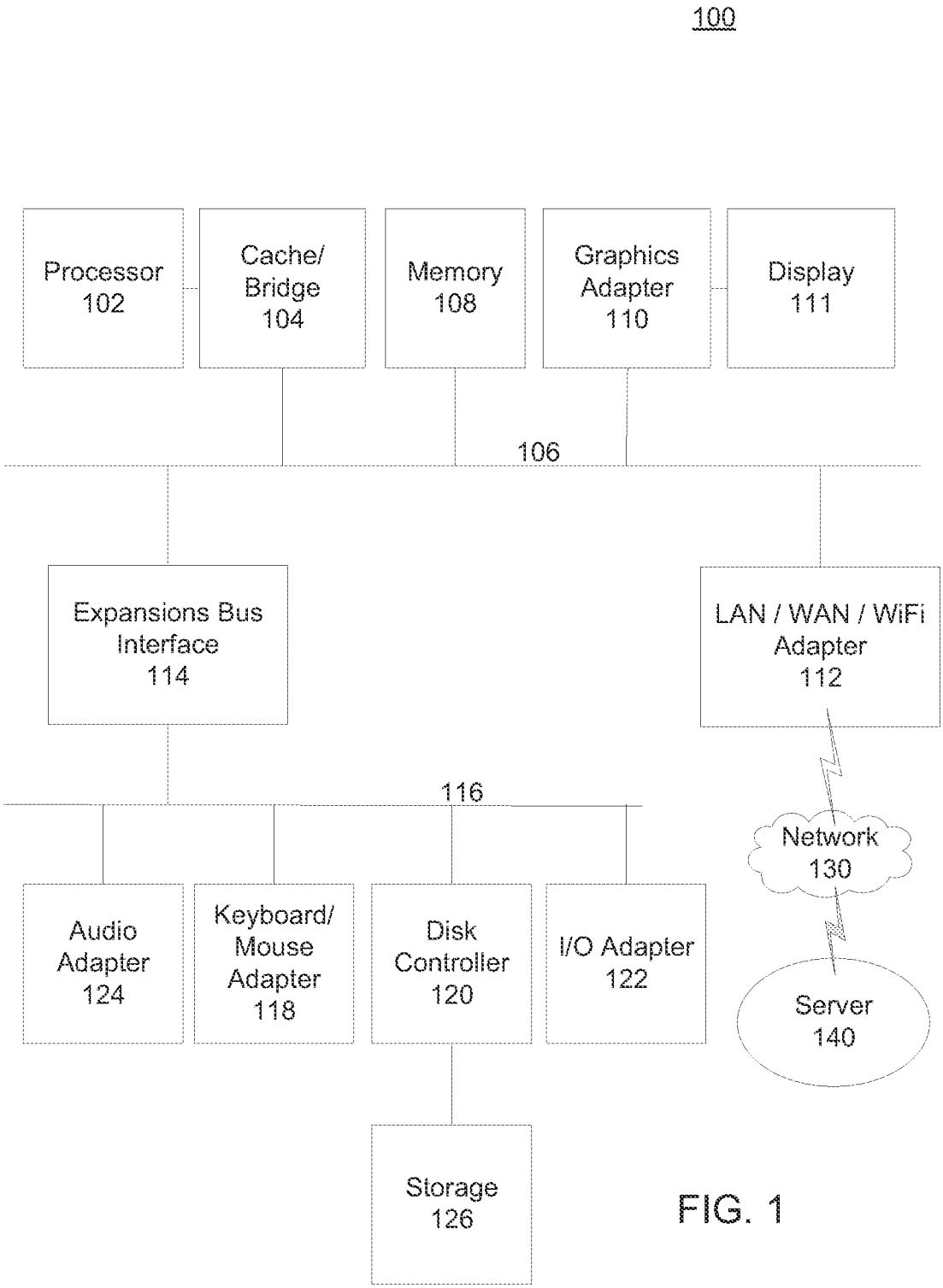
100
Processor 102
Cache/ Bridge 104
Memory 108
Graphics Adapter 110
Display 111
106
Expansions Bus Interface 114
LAN / WAN / WiFi Adapter 112
116
Network 130
Audio Adapter 124
Keyboard/ Mouse Adapter 118
Disk Controller 120
I/O Adapter 122
Server 140
Storage 126

FIG. 1

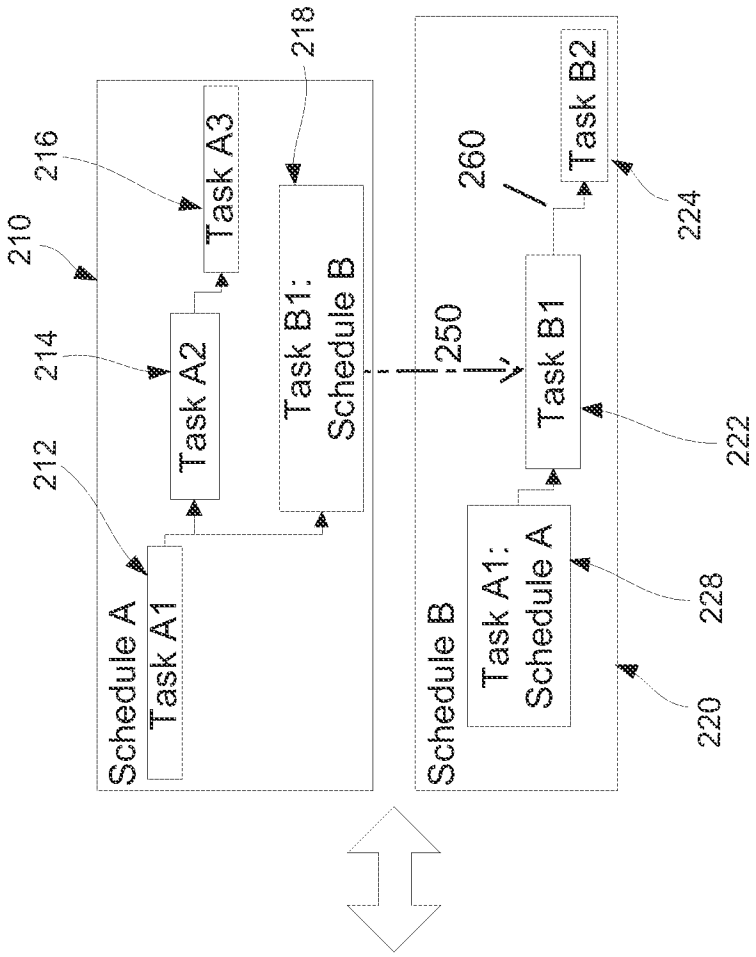
Schedule A
Task A1
Task A2
Task A3
Task B1: Schedule B
Schedule B
Task A1: Schedule A
Task B1
Task B2
210
212
214
216
218
220
222
224
228
250
260

FIG. 2B

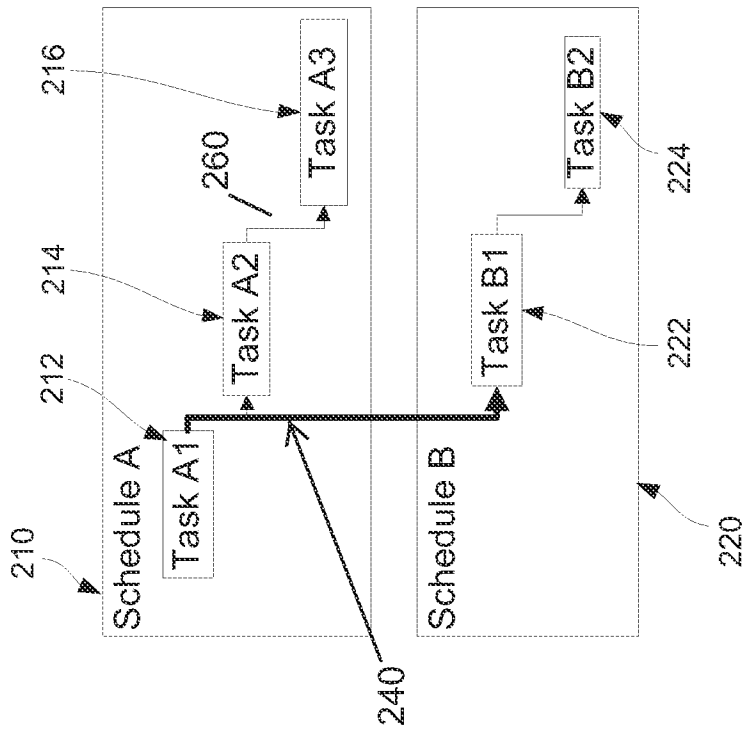
Schedule A
Task A1
Task A2
Task A3
Schedule B
Task B1
Task B2
210
212
214
216
220
222
224
240
260

FIG. 2A

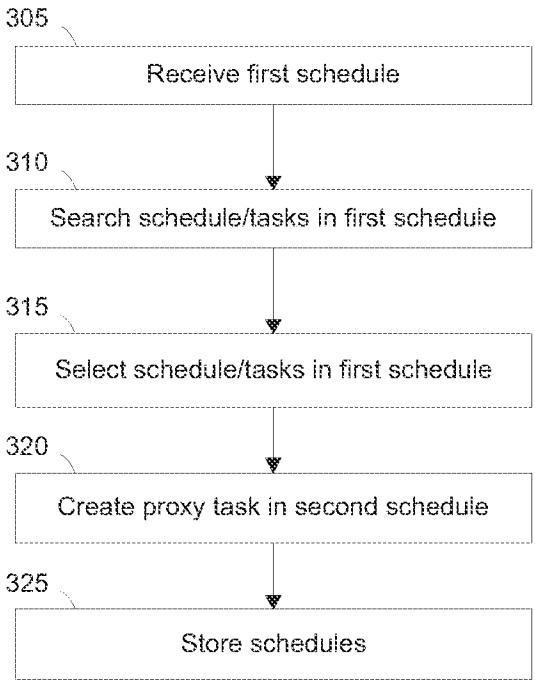
305
Receive first schedule
310
Search schedule/tasks in first schedule
315
Select schedule/tasks in first schedule
320
Create proxy task in second schedule
325
Store schedules

FIG. 3

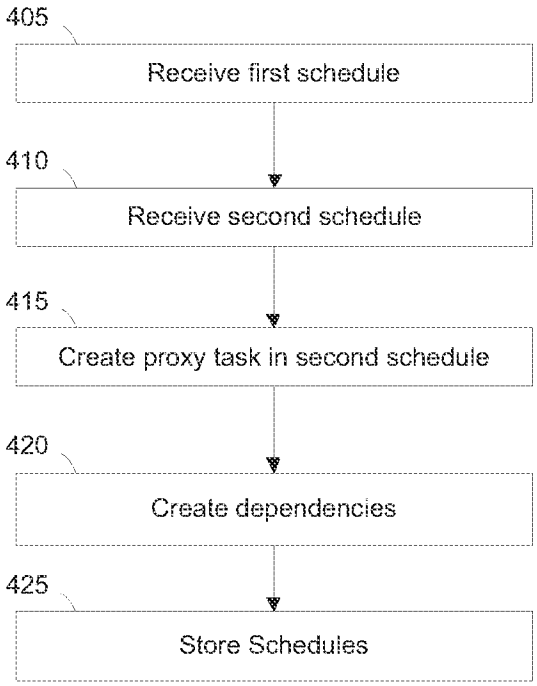
405
Receive first schedule
410
Receive second schedule
415
Create proxy task in second schedule
420
Create dependencies
425
Store Schedules

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2012/033180

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/48 G06Q10/00 G06F21/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F G06Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2004/078258 A1 (SCHULZ KARSTEN [AU] ET AL) 22 April 2004 (2004-04-22) claim 1 figures 1-3 paragraph [0002] -----	1-18
X	US 2002/040304 A1 (SHENOY SUBRAO [US] ET AL) 4 April 2002 (2002-04-04) paragraph [0226] -----	1-18
X	WO 2007/094816 A2 (APACHETA CORP [US]; HIGGINS STEVEN [US]; COLLINS JIM [US]; CLARE PETER) 23 August 2007 (2007-08-23) paragraph [0199] -----	1-18

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 June 2012	29/06/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Rackl, Günther

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2012/033180

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004078258 A1	22-04-2004	NONE	
US 2002040304 A1	04-04-2002	AU 9490501 A	15-04-2002
		US 2002040304 A1	04-04-2002
		WO 0229670 A1	11-04-2002
WO 2007094816 A2	23-08-2007	US 2007067373 A1	22-03-2007
		WO 2007094816 A2	23-08-2007