



(19) **United States**

(12) **Patent Application Publication**
Charisius et al.

(10) **Pub. No.: US 2002/0075293 A1**

(43) **Pub. Date: Jun. 20, 2002**

(54) **METHODS AND SYSTEMS FOR ANIMATING
A WORKFLOW AND A PROJECT PLAN**

provisional application No. 60/296,707, filed on Jun. 7, 2001.

(76) Inventors: **Dietrich Charisius, Stuttgart (DE);
Peter Coad, Raleigh, NC (US)**

Publication Classification

(51) **Int. Cl.⁷ G06F 3/00**
(52) **U.S. Cl. 345/704**

Correspondence Address:
**SONNENSCHN NATH & ROSENTHAL
P.O. BOX 061080
WACKER DRIVE STATION
CHICAGO, IL 60606-1080 (US)**

(57) **ABSTRACT**

Methods and systems consistent with the present invention allow a user to animate different versions of a plan or workflow. Each version reflects an instance in an edit history, i.e., reflects the changes made to the plan or workflow. Additionally, methods and systems consistent with the present invention allow a user to view the various plans created from a given workflow over time. Finally, methods and systems consistent with the present invention may be used to review the steps performed during the activation of a plan.

(21) Appl. No.: **09/944,696**

(22) Filed: **Aug. 31, 2001**

Related U.S. Application Data

(63) Non-provisional of provisional application No. 60/230,054, filed on Sep. 1, 2000. Non-provisional of

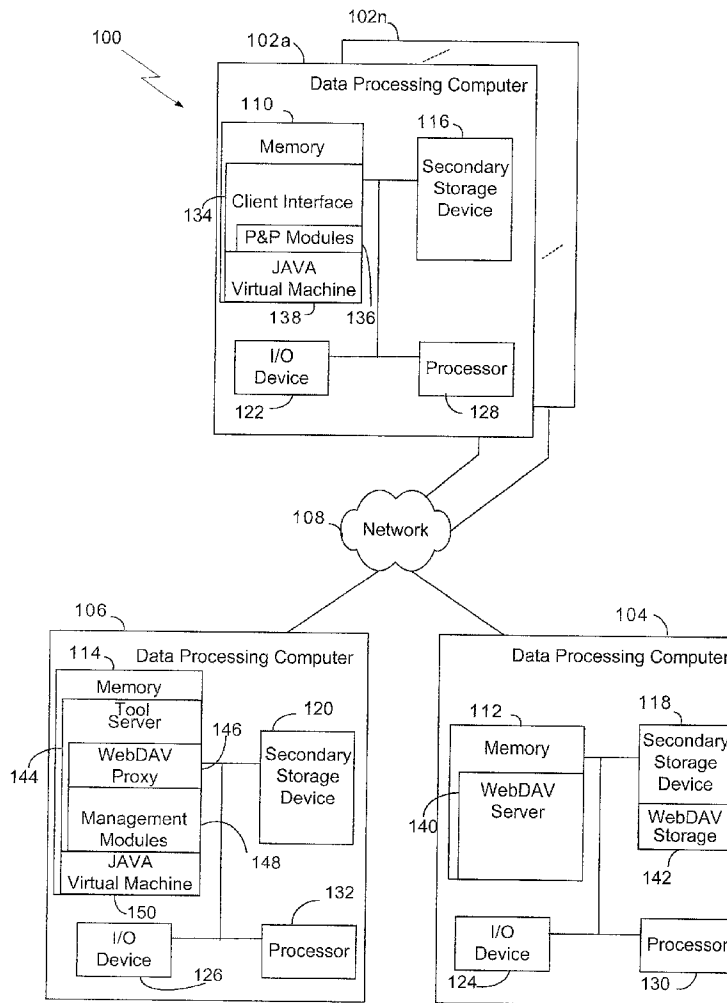


FIG. 1

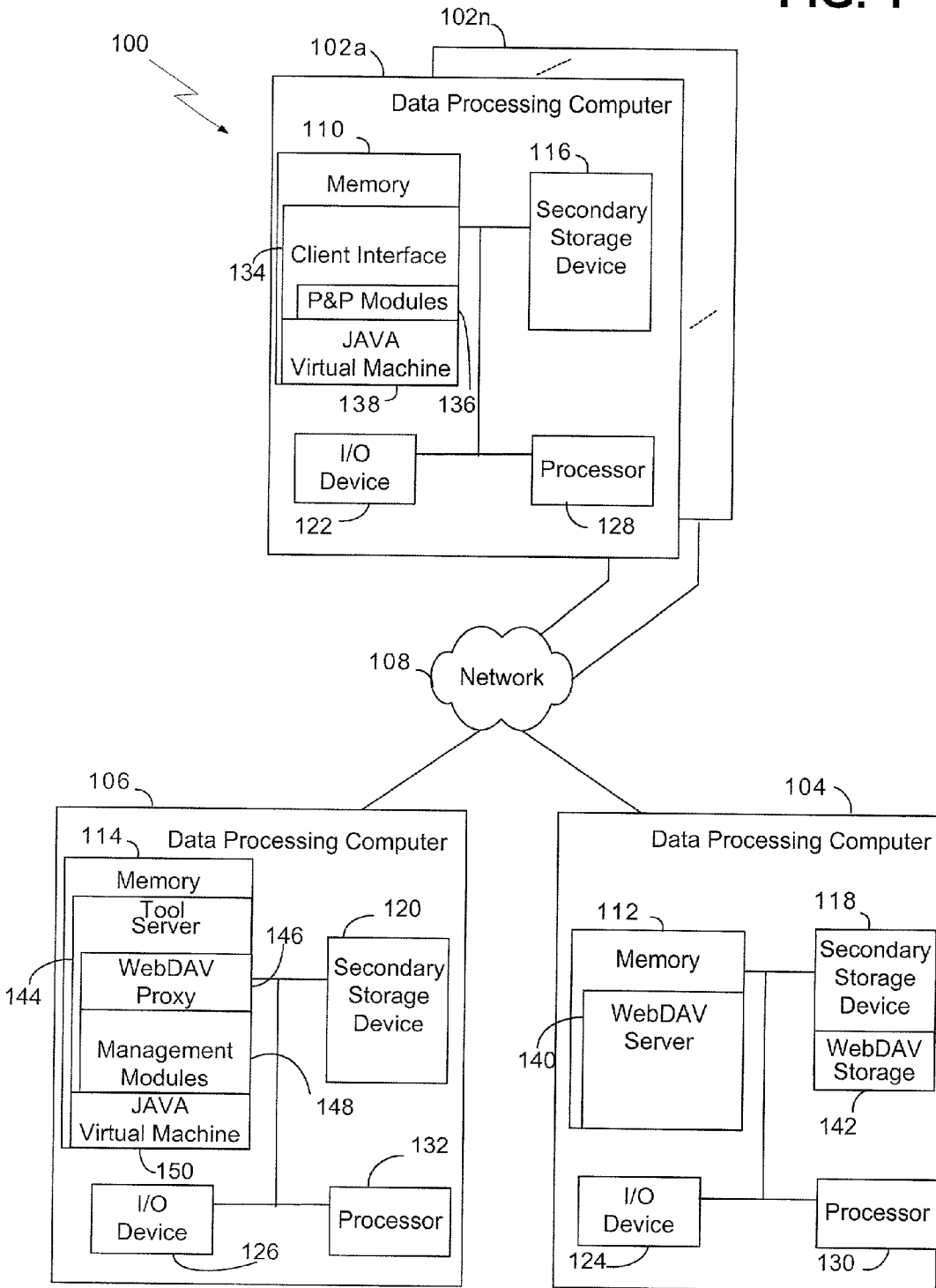


FIG. 2

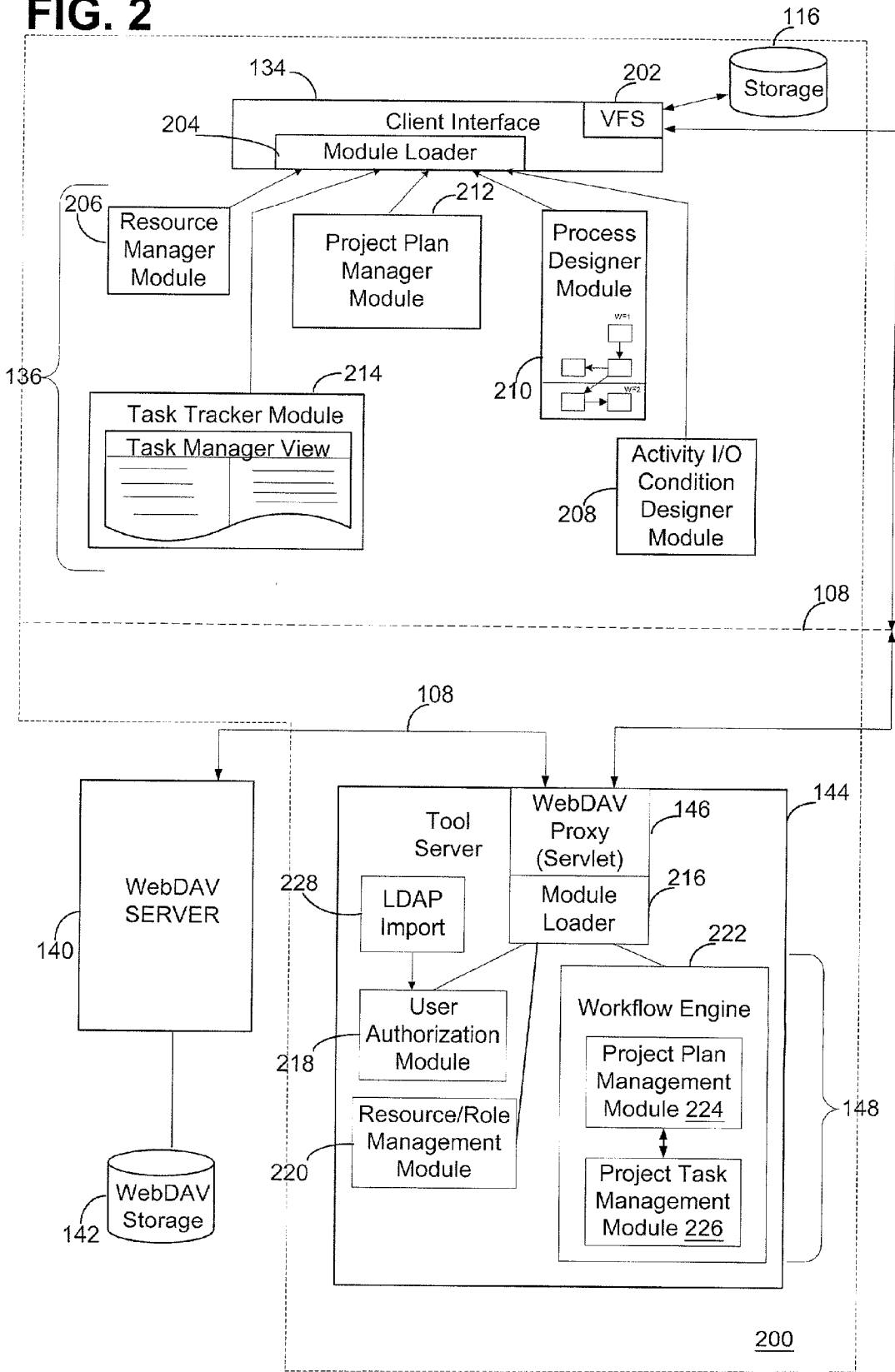


FIG. 3

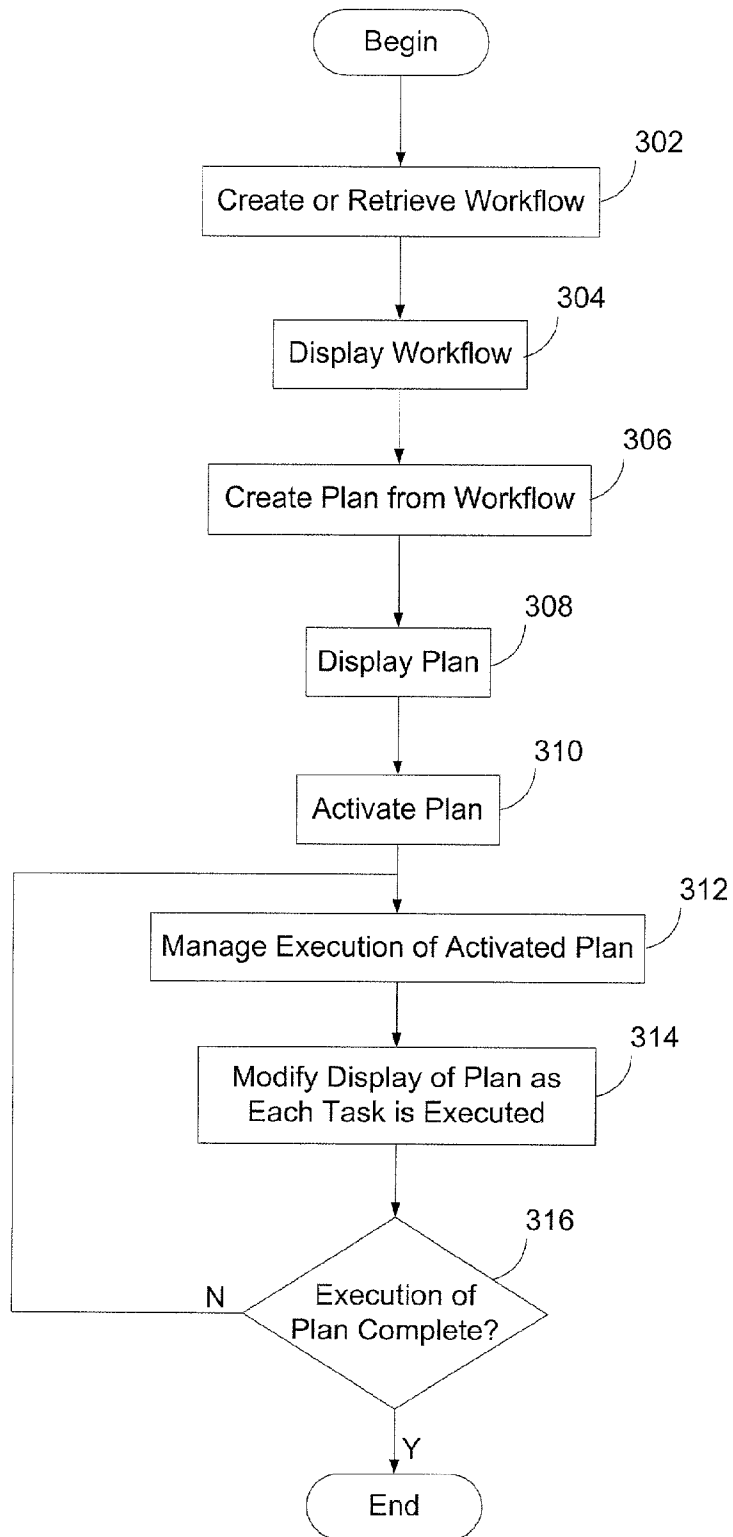


FIG. 4

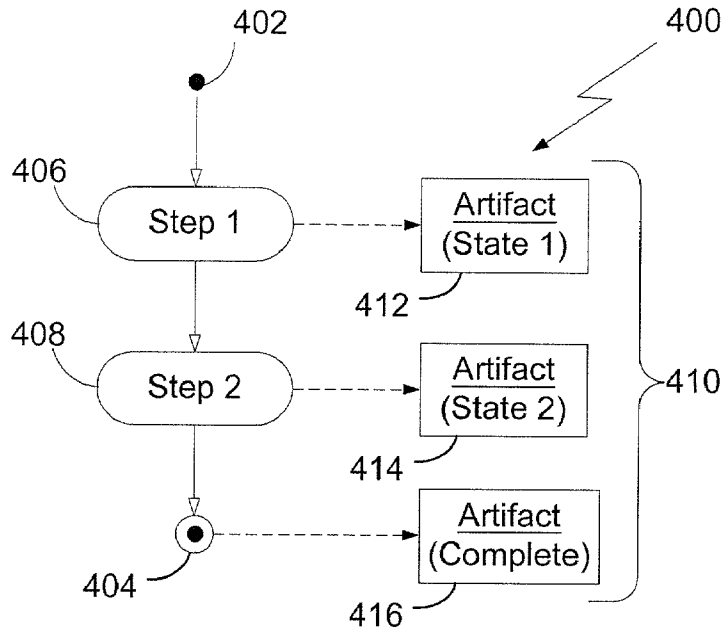


FIG. 5

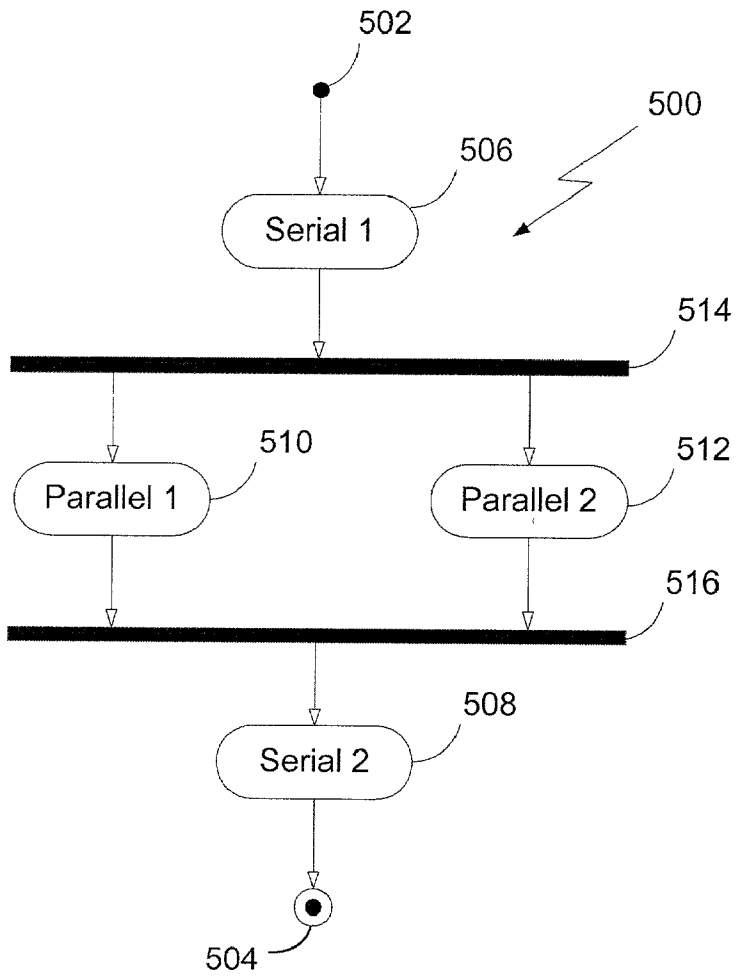


FIG. 6

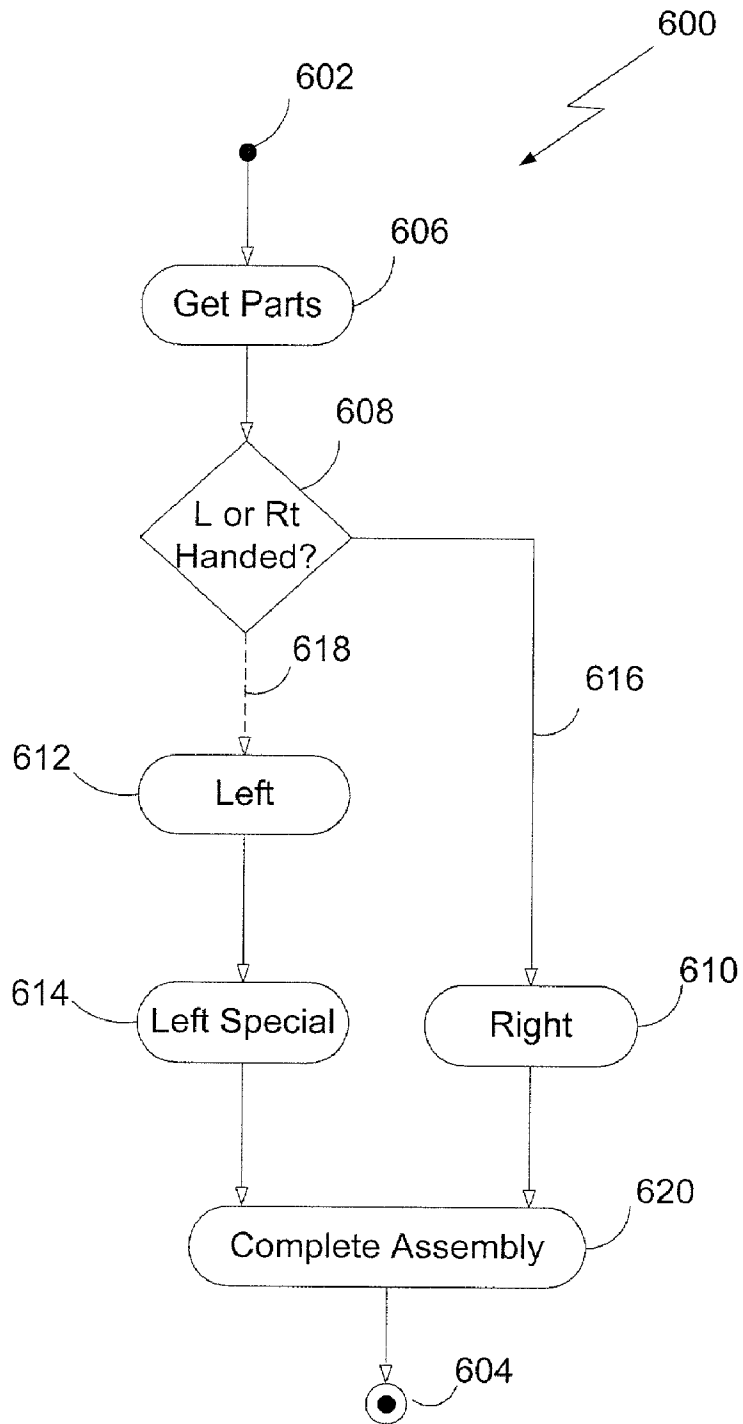


FIG. 7

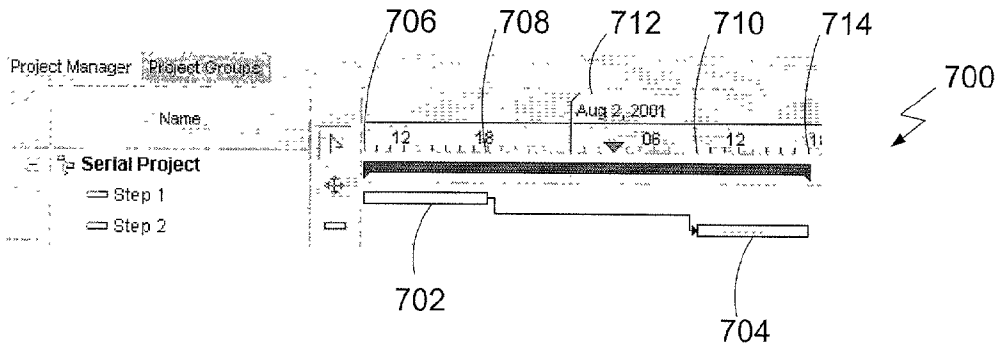


FIG. 8

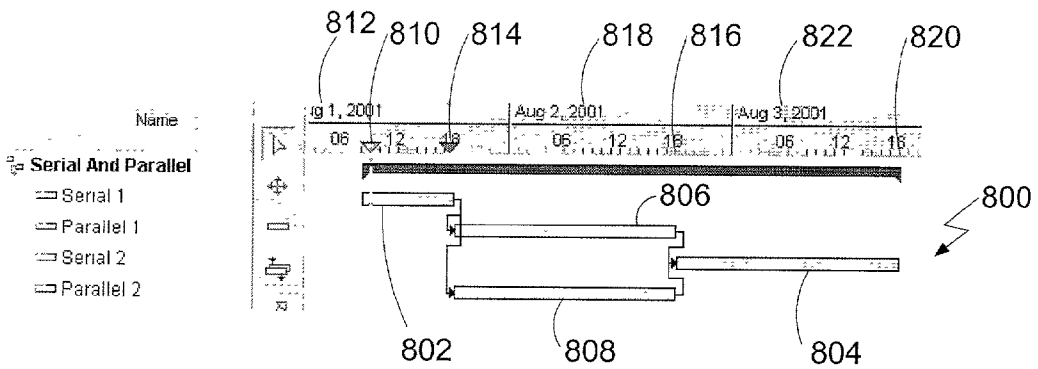


FIG. 9

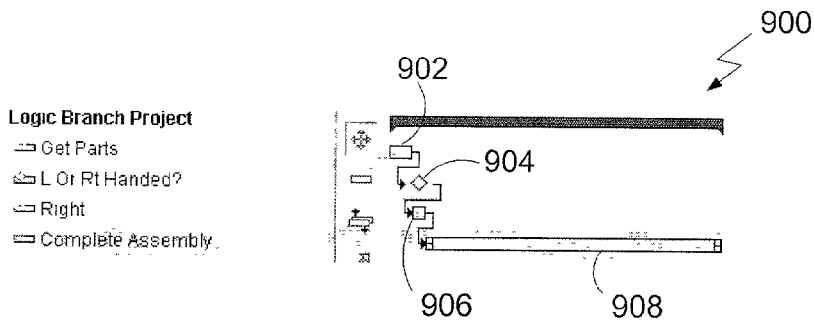


FIG. 10

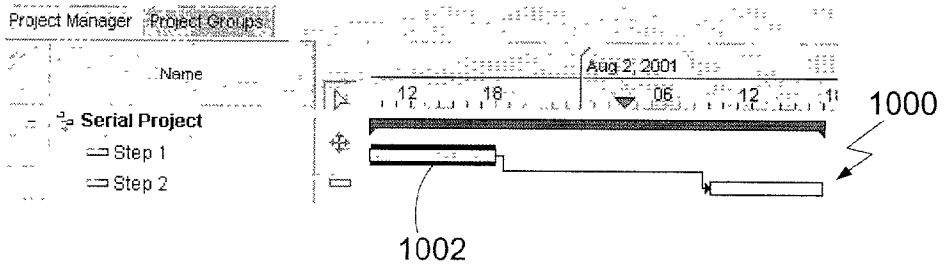


FIG. 11

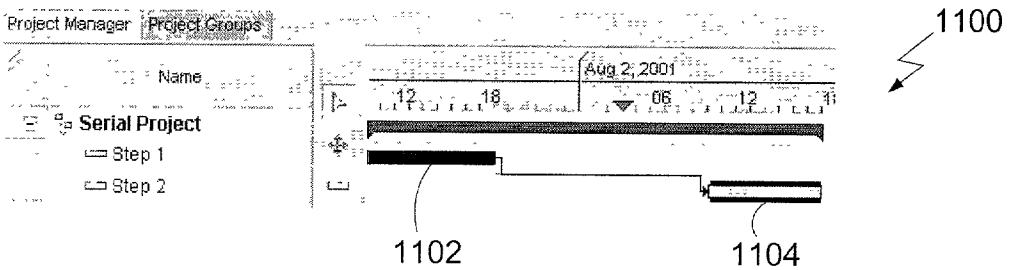


FIG. 12

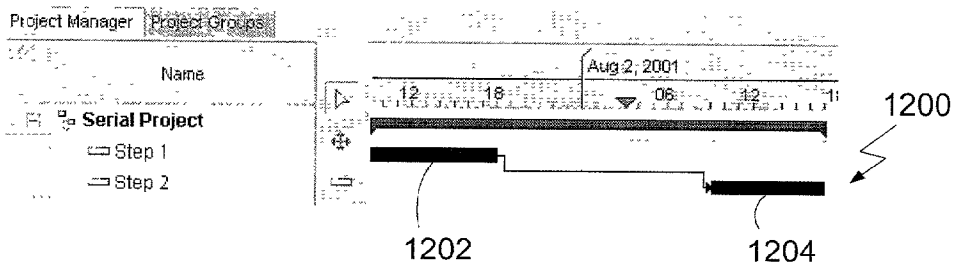


FIG. 13

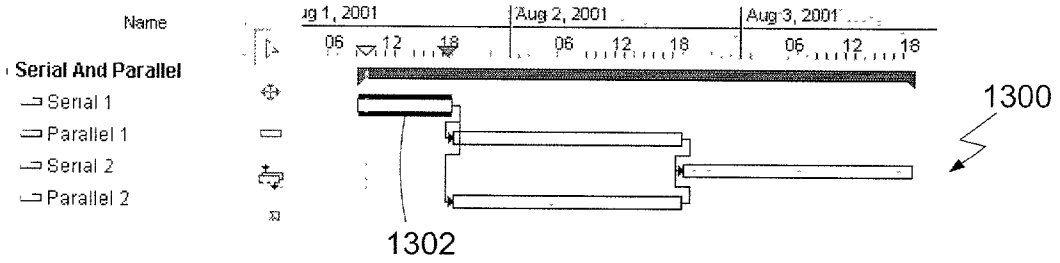


FIG. 14

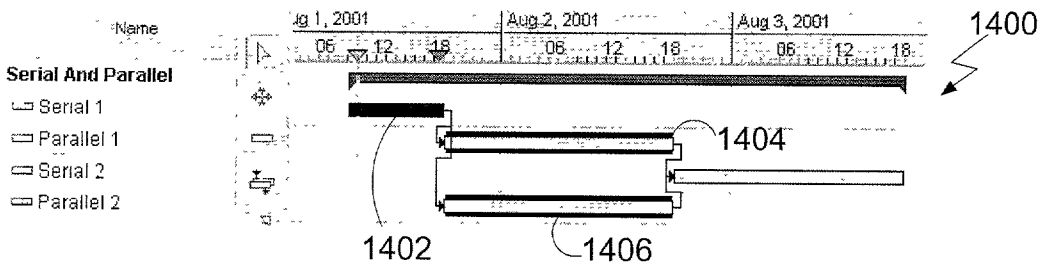


FIG. 15

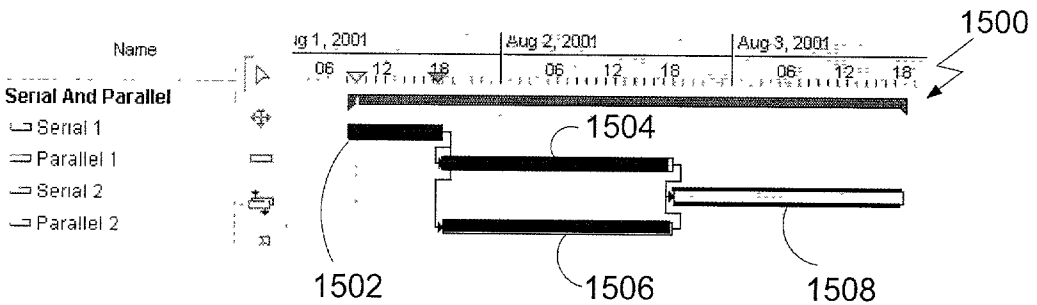


FIG. 16

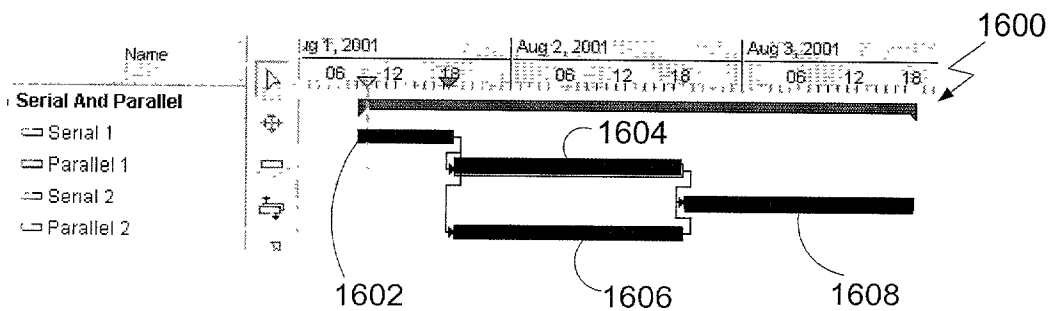


FIG. 17

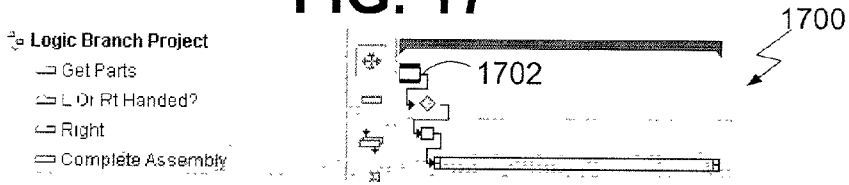


FIG. 18

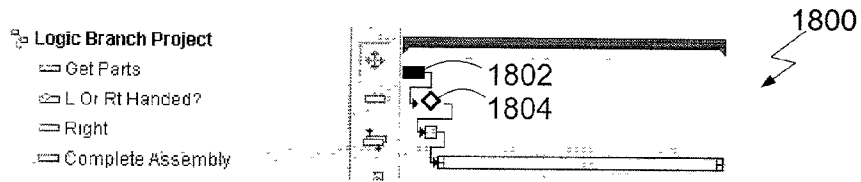


FIG. 19

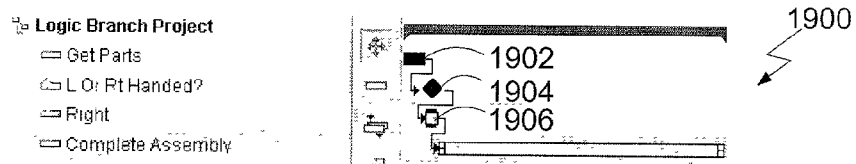


FIG. 20

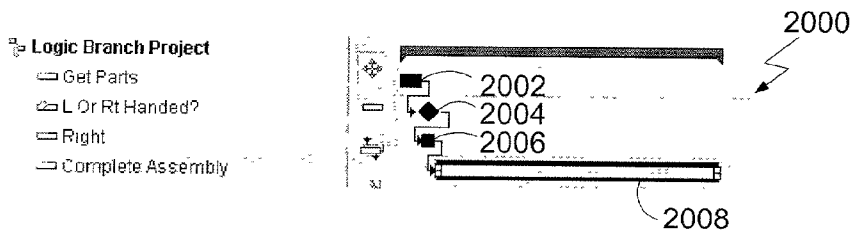


FIG. 21

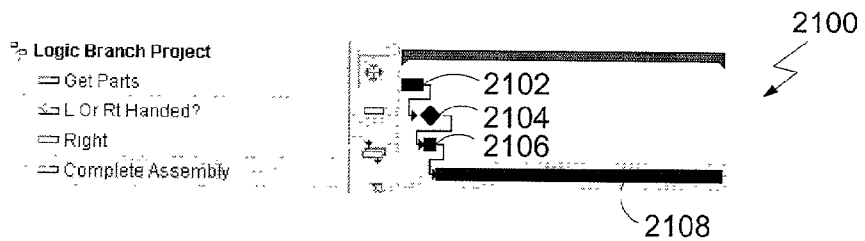


FIG. 22

- Logic Branch Project
 - Get Parts
 - L Or Rt Handed?
 - Right
 - Complete Assembly

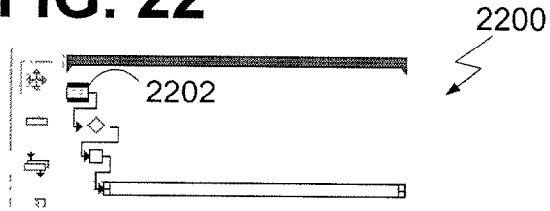


FIG. 23

- Logic Branch Project
 - Get Parts
 - L Or Rt Handed?
 - Right
 - Complete Assembly

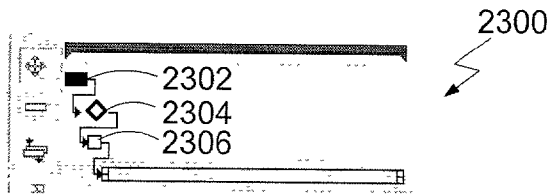


FIG. 24

- Logic Branch
 - Get Parts
 - L Or Rt Handed?
 - Left
 - Left Special
 - Complete Assembly

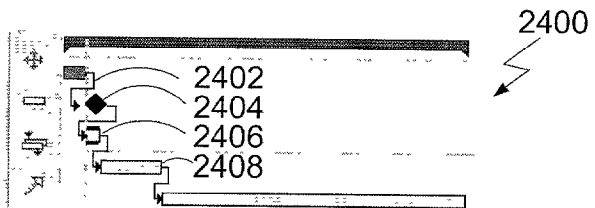


FIG. 25

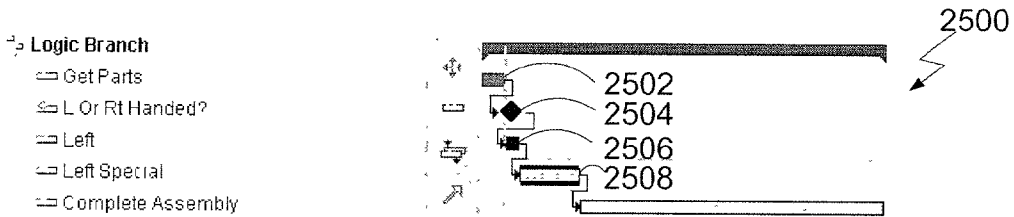


FIG. 26

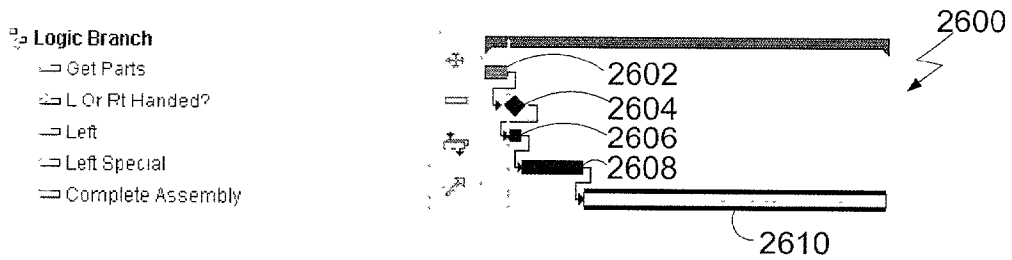


FIG. 27

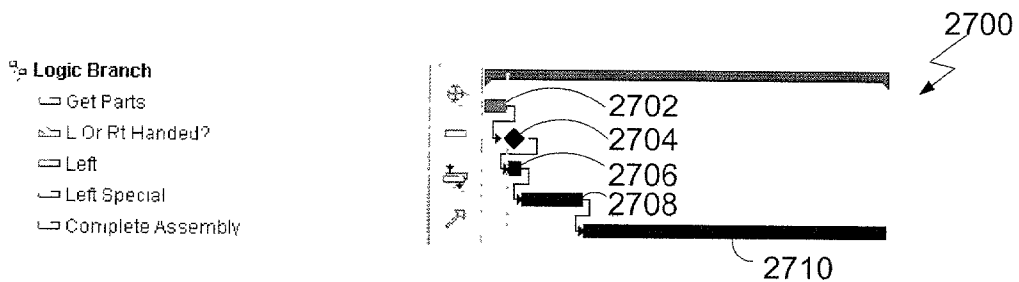


FIG. 28A

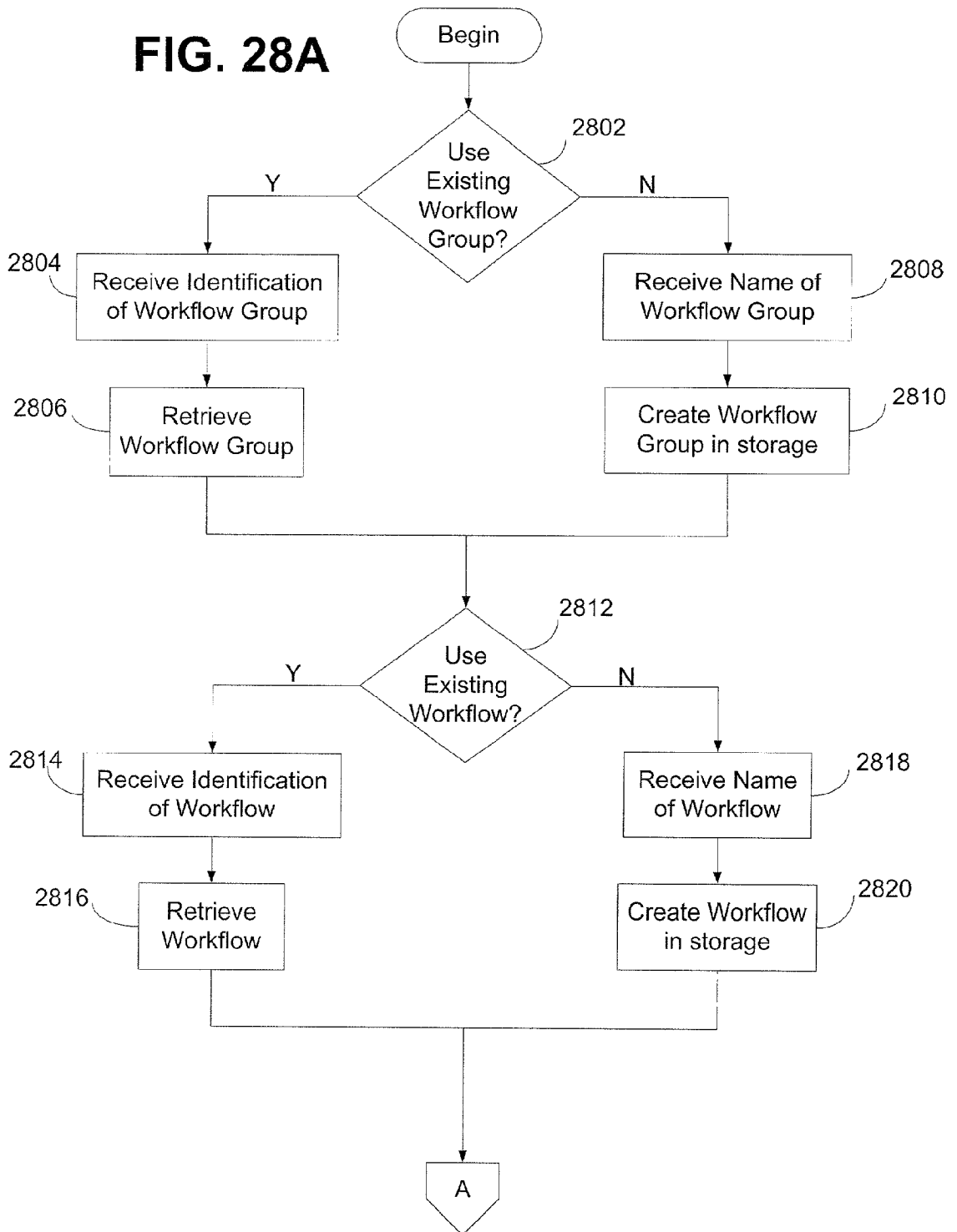


FIG. 28B

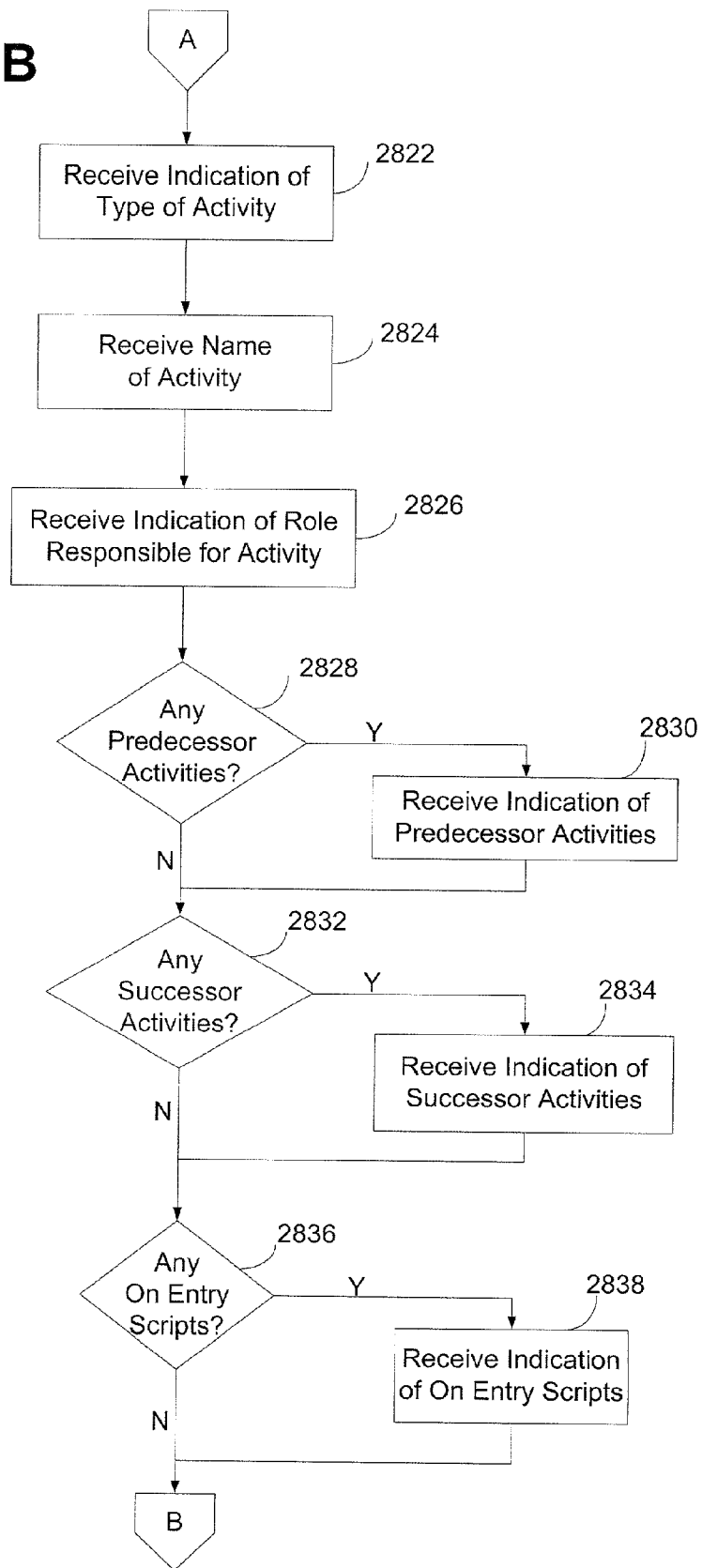


FIG. 28C

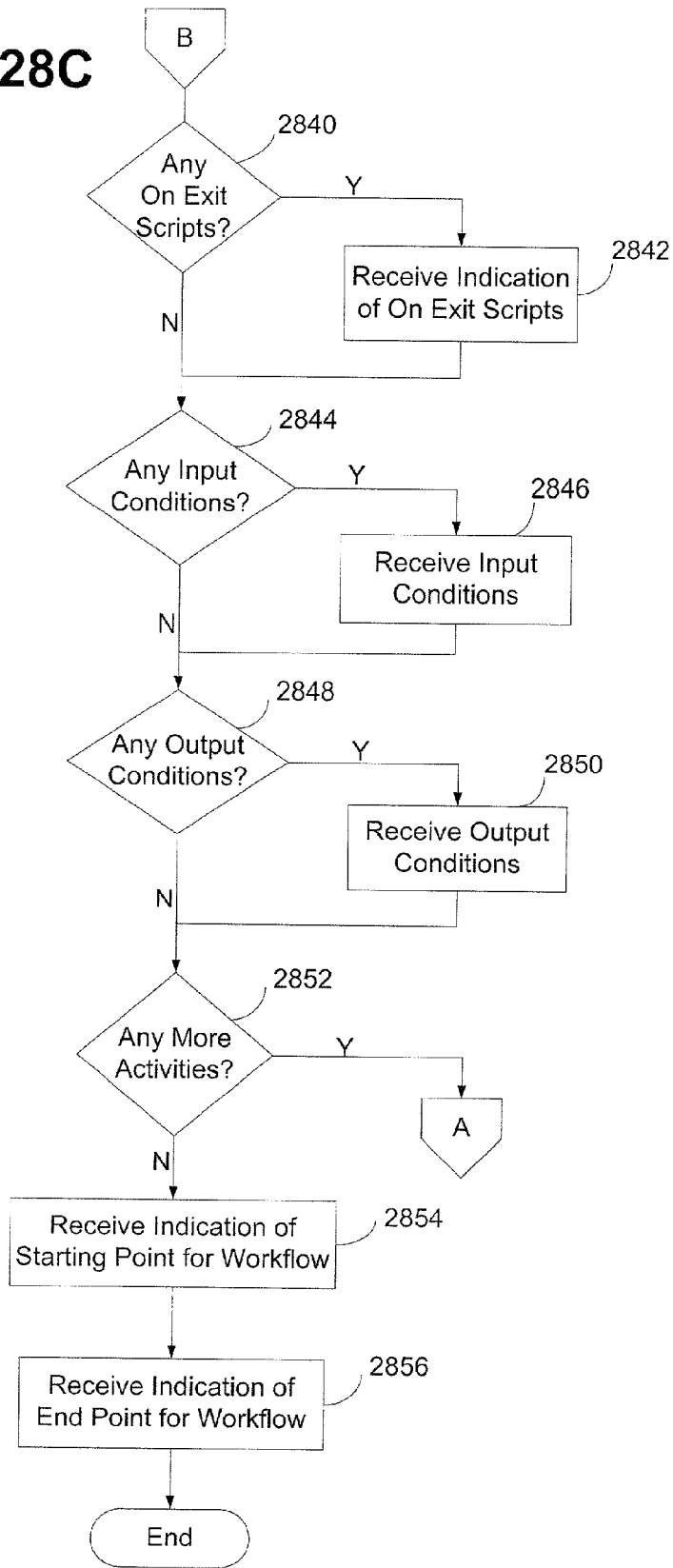


FIG. 29

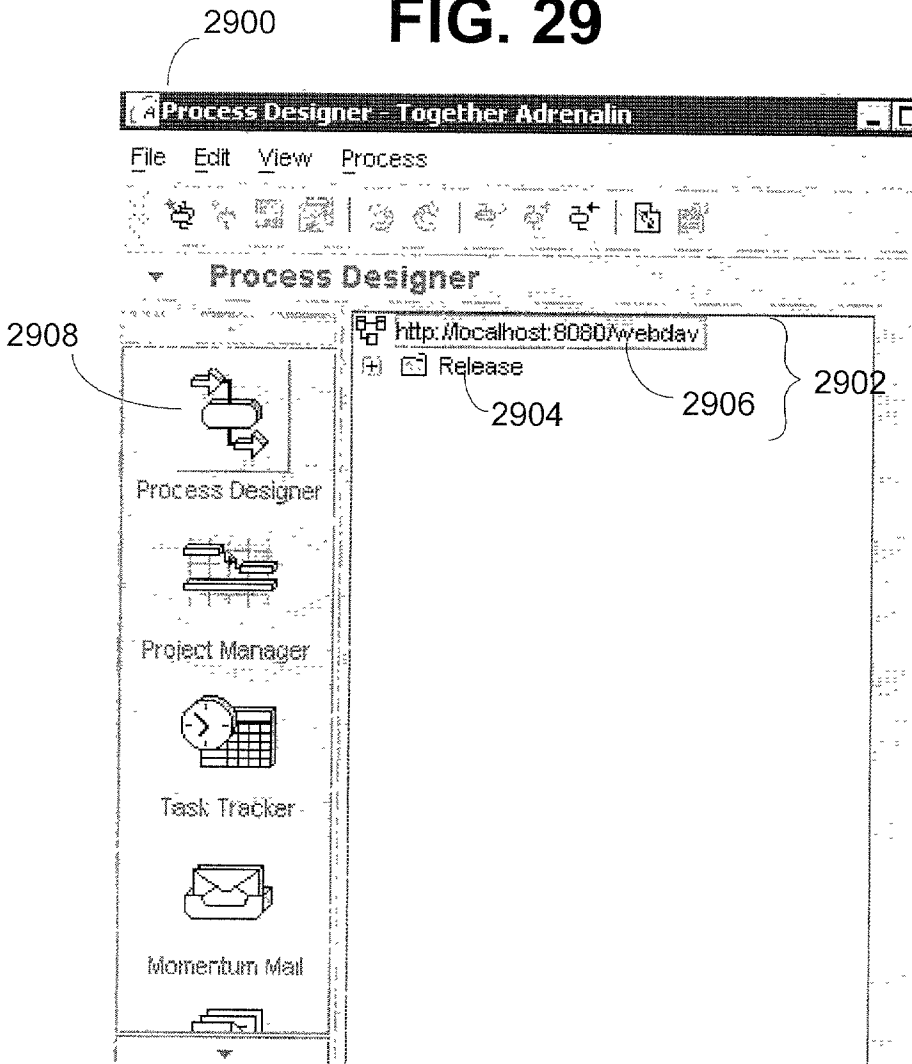


FIG. 30

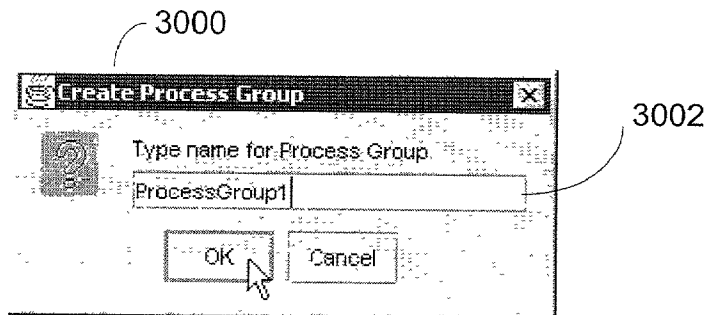


FIG. 31

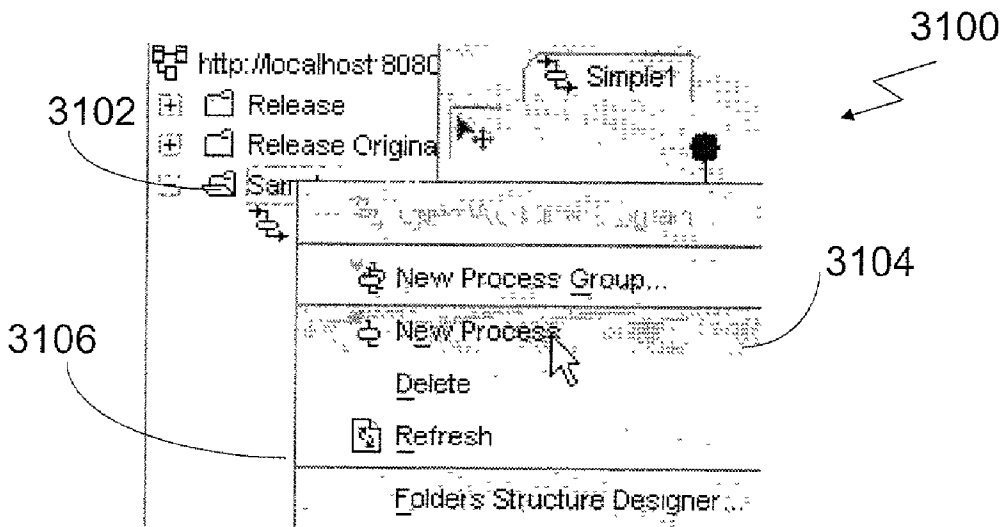


FIG. 32

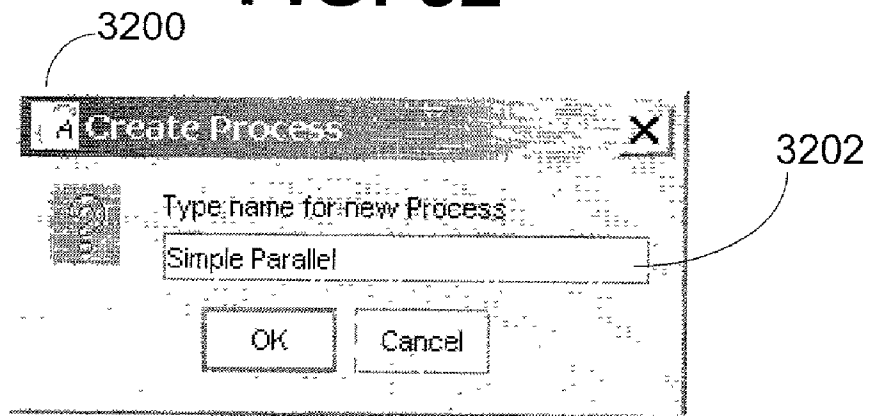


FIG. 33A

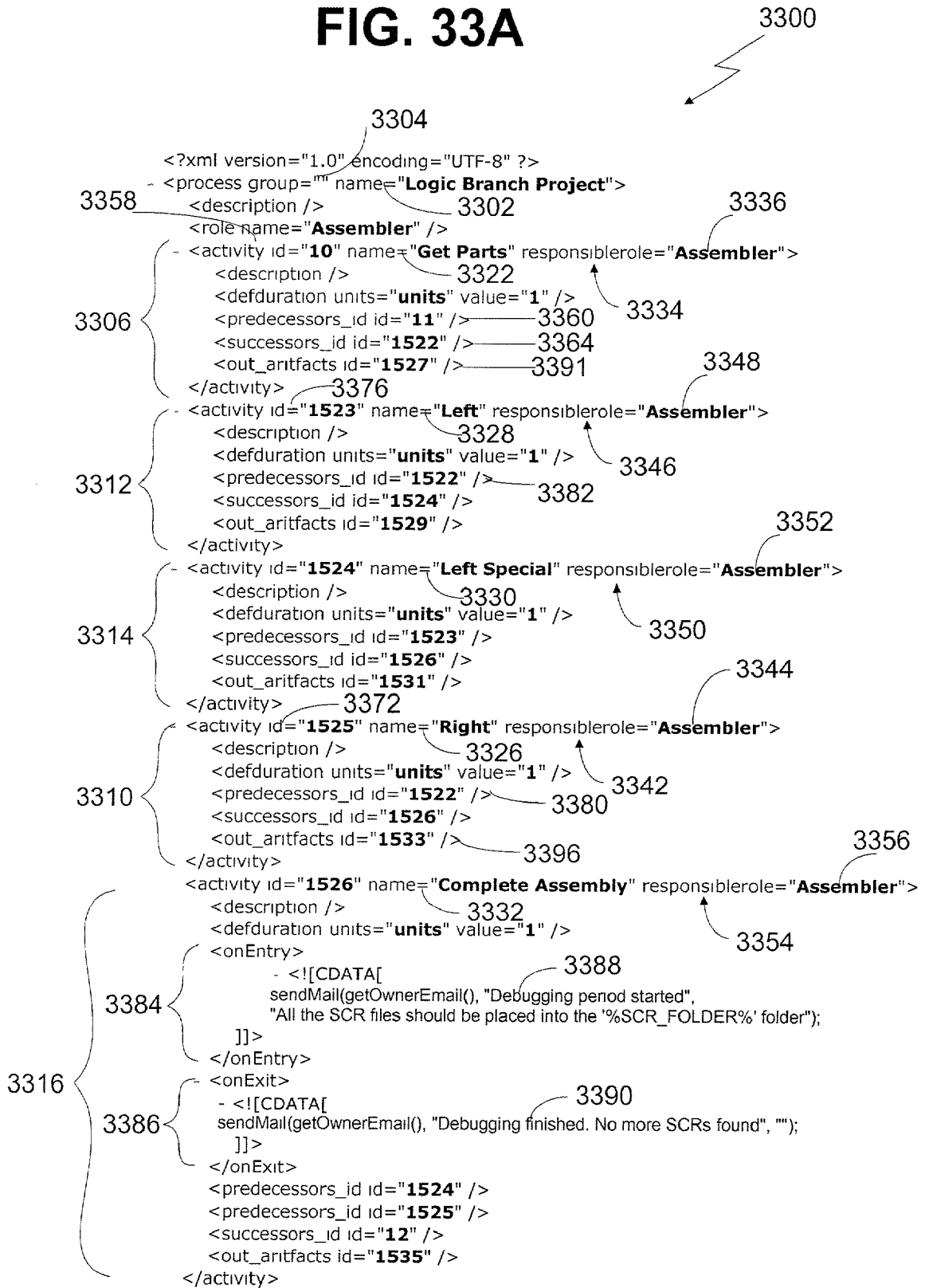


FIG. 33B

```

3394 {
- <artifact id="1527" identity="linkable1" name="Document is INITIAL">
  <description>The condition becomes true when the %Document% artifact gets the
    INITIAL state. To make this condition valid you should define the %Document%
    parameter and optionally define the document template and the application to
    open the document.</description>
  <type>linkable</type>
  <link link="%Document%" />
  <state>INITIAL</state>
  <event id="1528" />
</artifact>
- <artifact id="1529" identity="linkable1" name="Document is LEFT">
  <description>The condition becomes true when the %Document% artifact gets the
    LEFT state. To make this condition valid you should define the %Document%
    parameter and optionally define the document template and the application to
    open the document.</description>
  <type>linkable</type>
  <link link="%Document%" />
  <state>LEFT</state>
  <event id="1530" />
</artifact>
- <artifact id="1531" identity="linkable1" name="Document is LEFT SPECIAL">
  <description>The condition becomes true when the %Document% artifact gets the
    LEFT SPECIAL state. To make this condition valid you should define the
    %Document% parameter and optionally define the document template and the
    application to open the document.</description>
  <type>linkable</type>
  <link link="%Document%" />
  <state>LEFT SPECIAL</state>
  <event id="1532" />
</artifact>
- <artifact id="1533" identity="linkable1" name="Document is RIGHT">
  <description>The condition becomes true when the %Document% artifact gets the
    RIGHT state. To make this condition valid you should define the %Document%
    parameter and optionally define the document template and the application to
    open the document.</description>
  <type>linkable</type>
  <link link="%Document%" />
  <state>RIGHT</state>
  <event id="1534" />
</artifact>
- <artifact id="1535" identity="linkable1" name="Document is APPROVED">
  <description>The condition becomes true when the %Document% artifact gets the
    APPROVED state. To make this condition valid you should define the
    %Document% parameter and optionally define the document template and the
    application to open the document.</description>
  <type>linkable</type>
  <link link="%Document%" />
  <state>APPROVED</state>
  <event id="1536" />
</artifact>
}
  
```

3392 points to the first artifact's name attribute.

3393 points to the first artifact's description.

3395 points to the first artifact's event id attribute.

3397 points to the third artifact's name attribute.

3398 points to the third artifact's description.

3399 points to the fourth artifact's event id attribute.

FIG. 33C

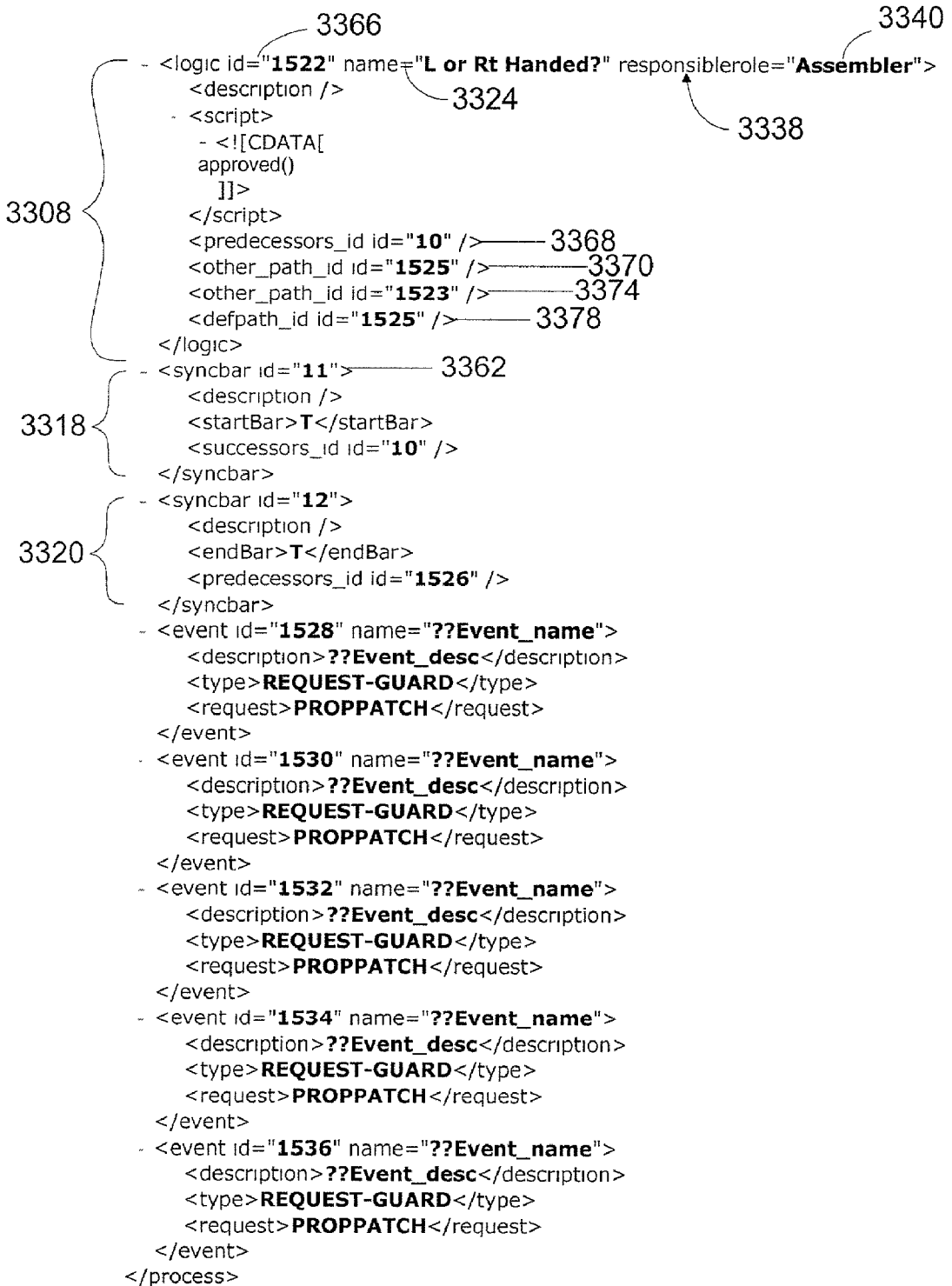


FIG. 34

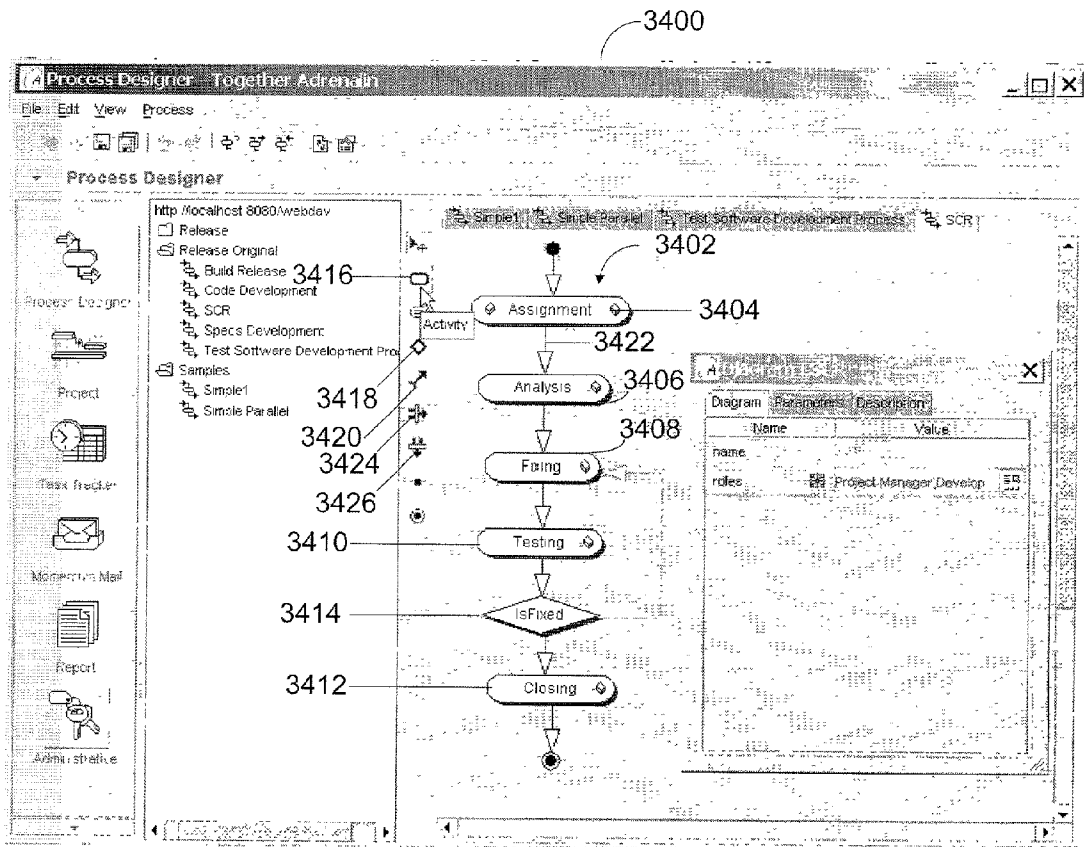


FIG. 35

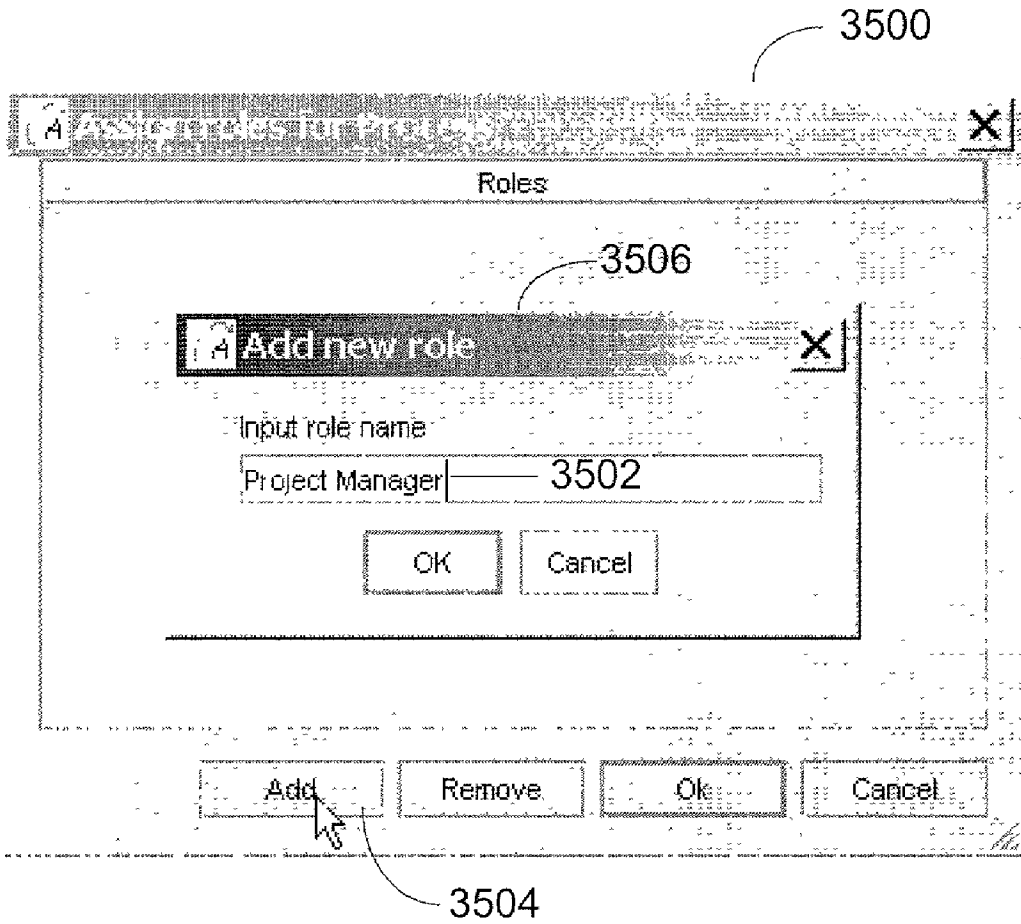


FIG. 36

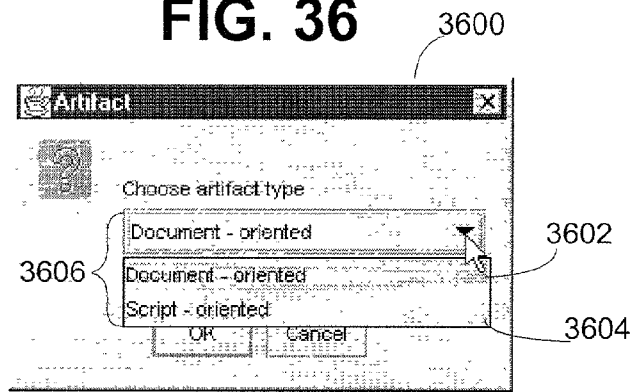


FIG. 37

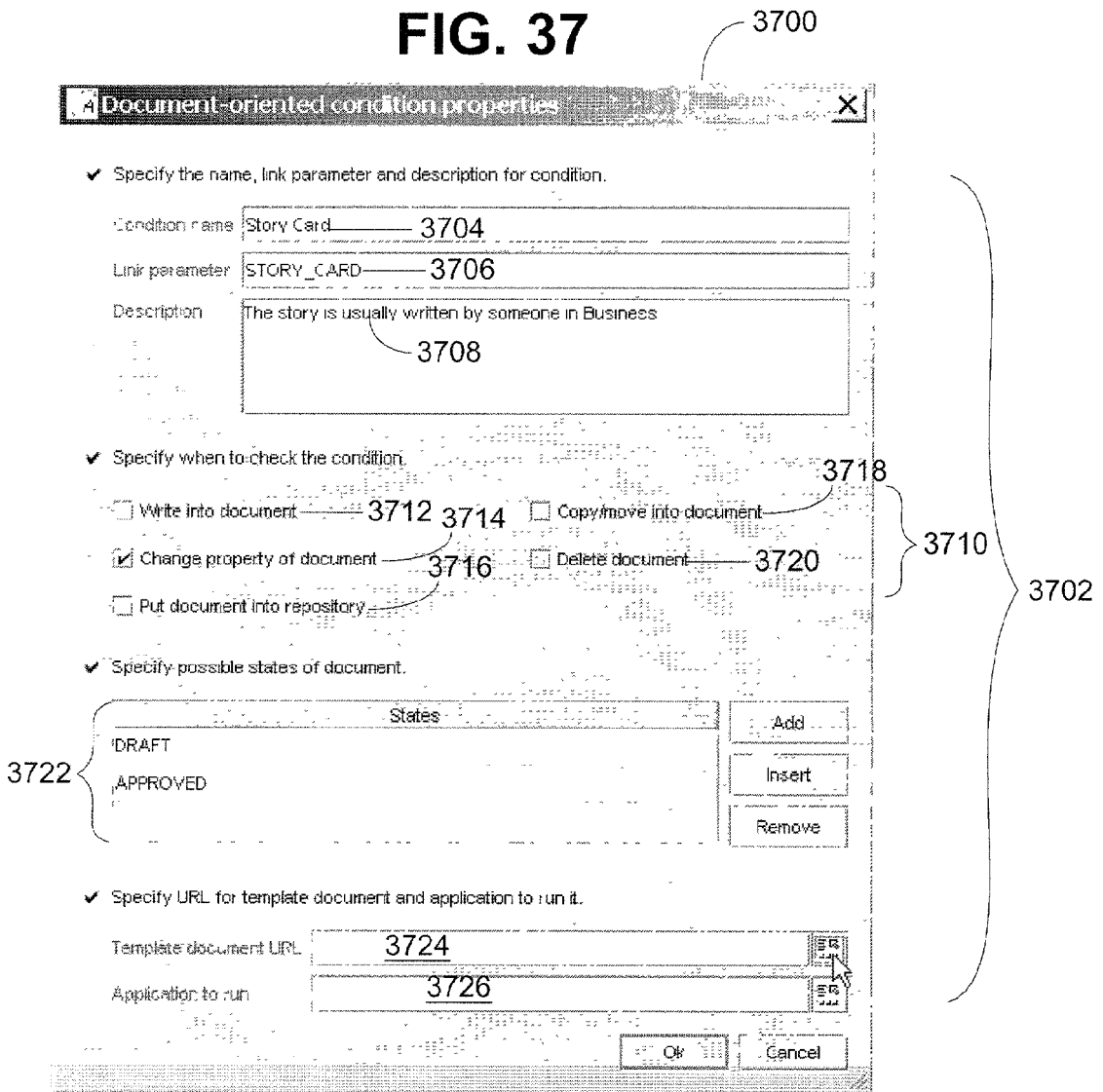


FIG. 38

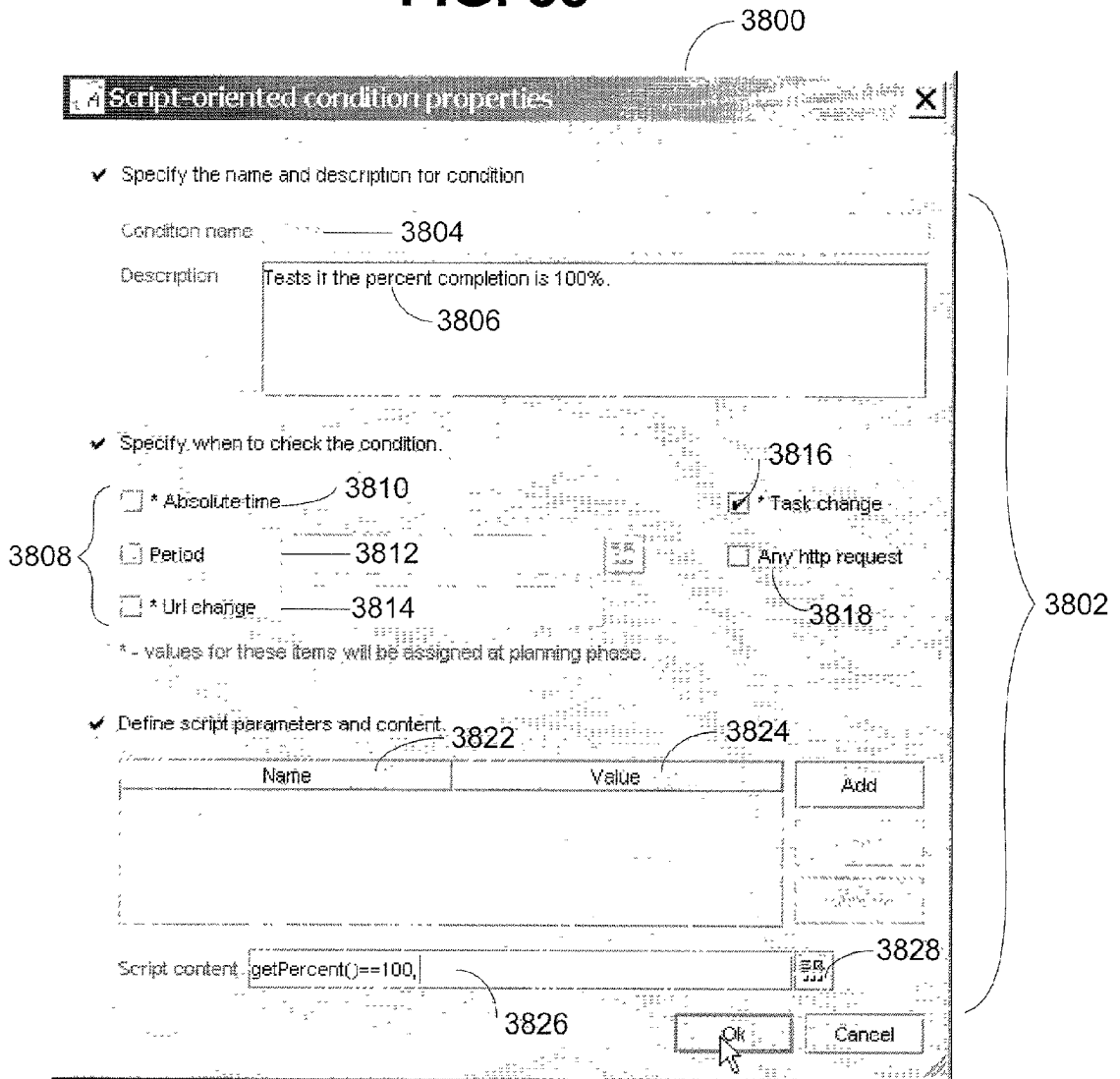


FIG. 39

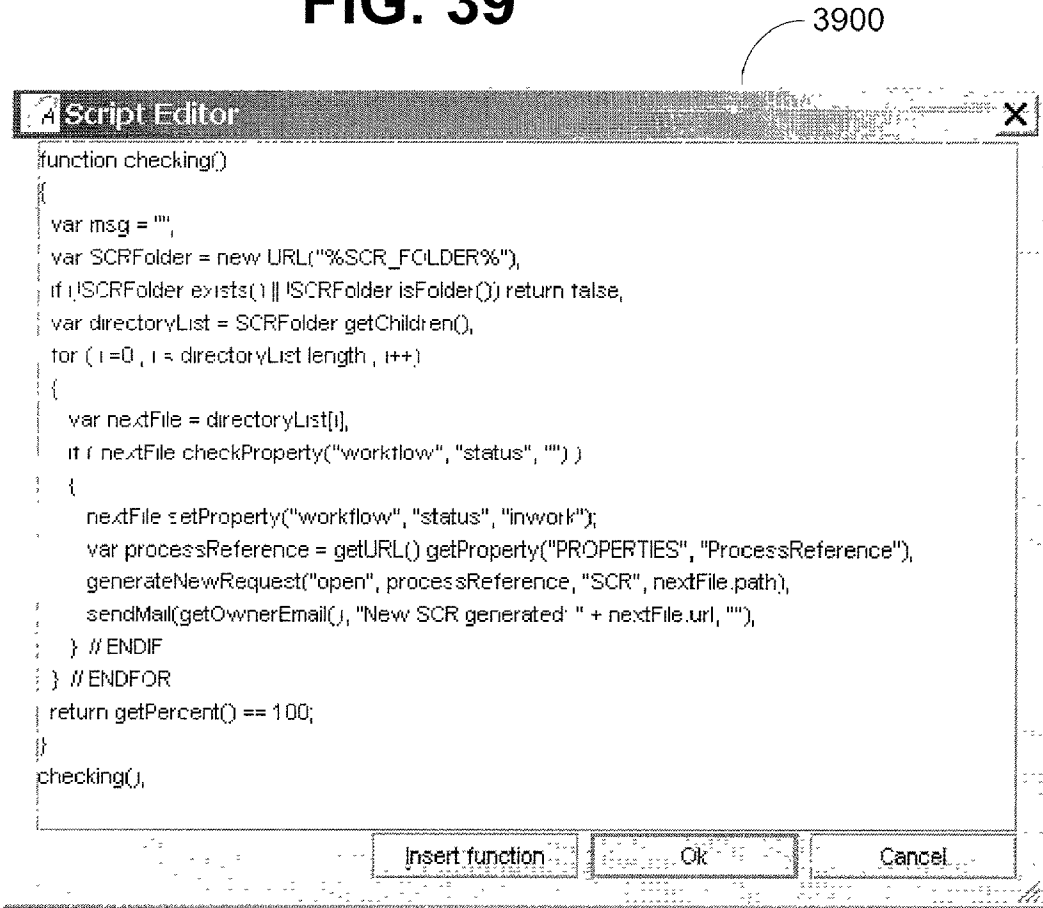


FIG. 40

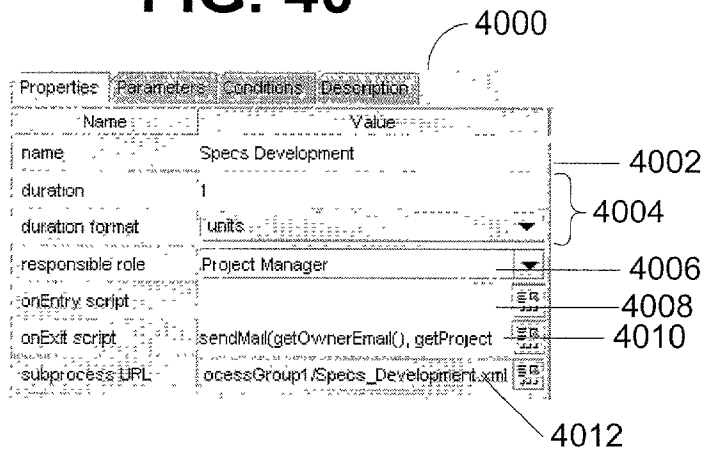


FIG. 41A

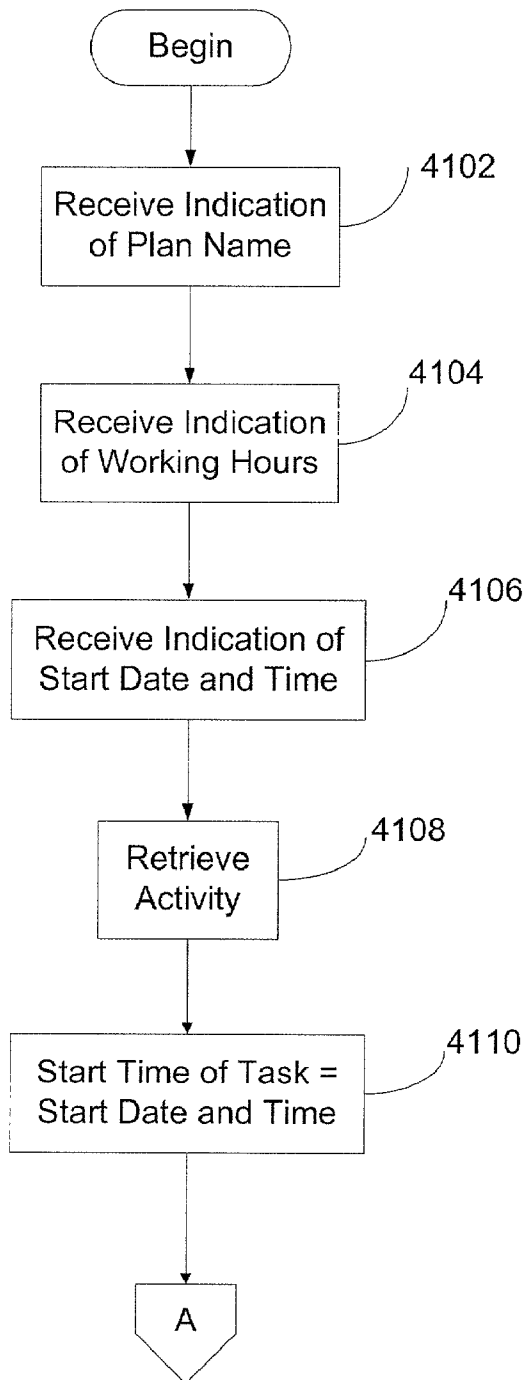


FIG. 41B

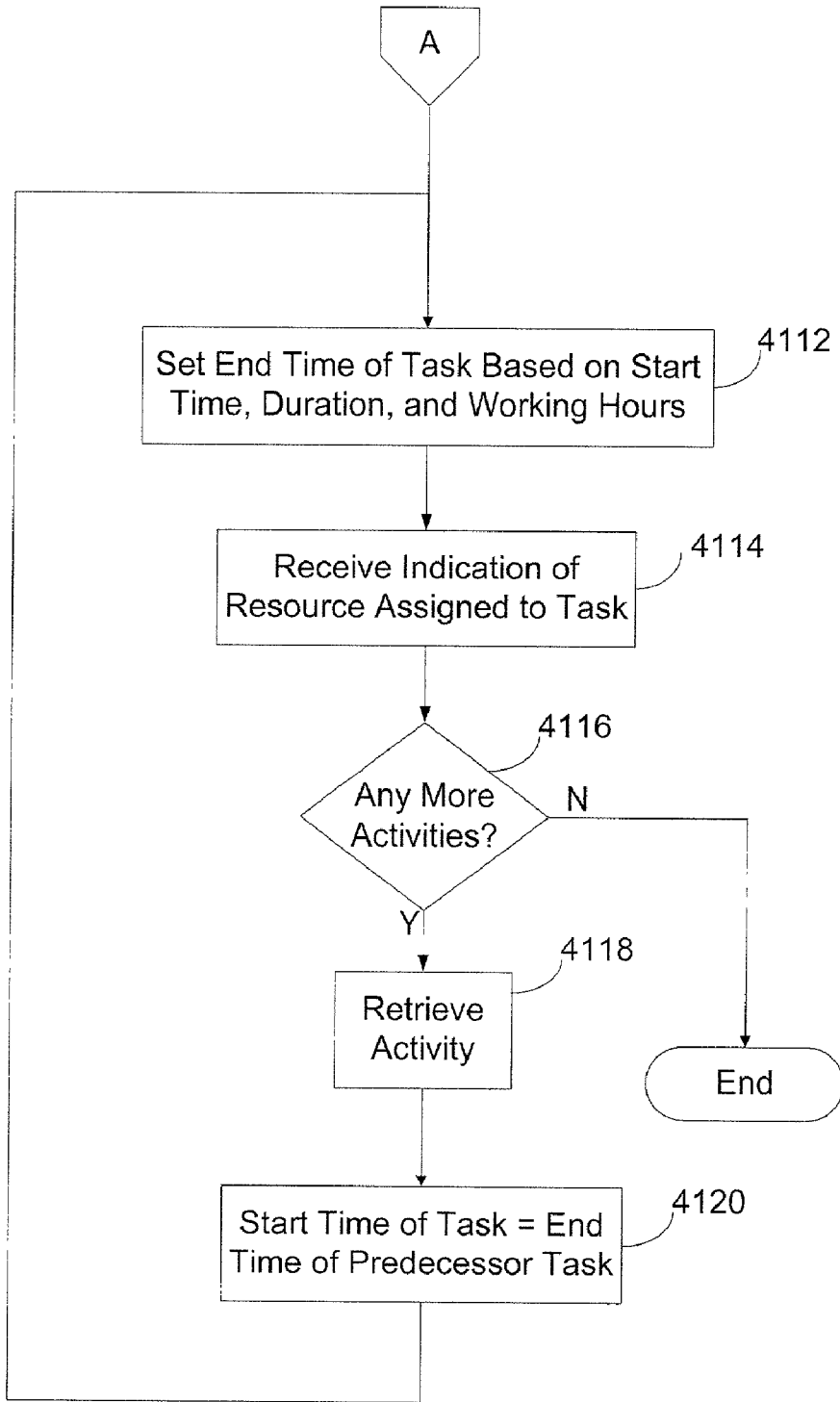


FIG. 42

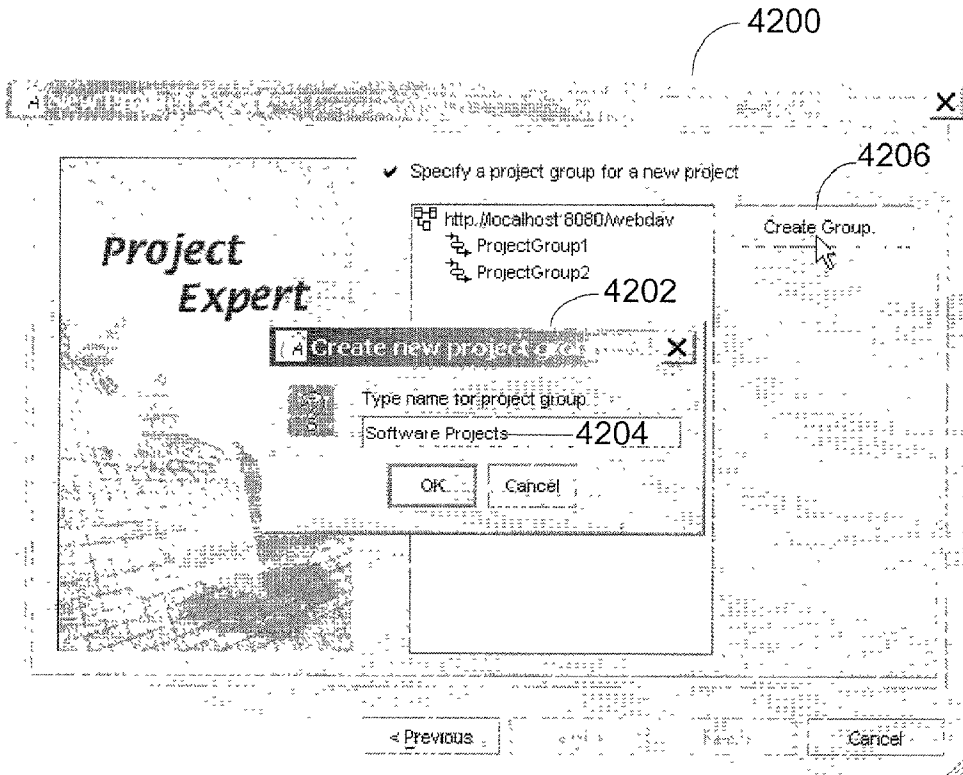


FIG. 43

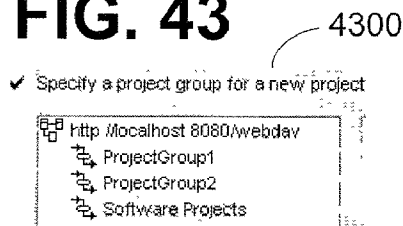


FIG. 44

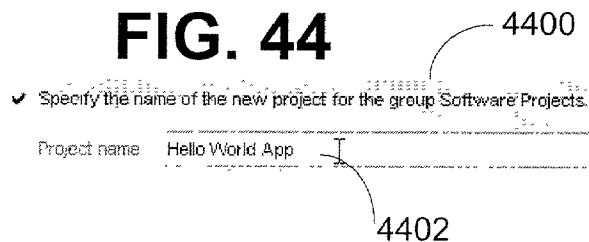


FIG. 45

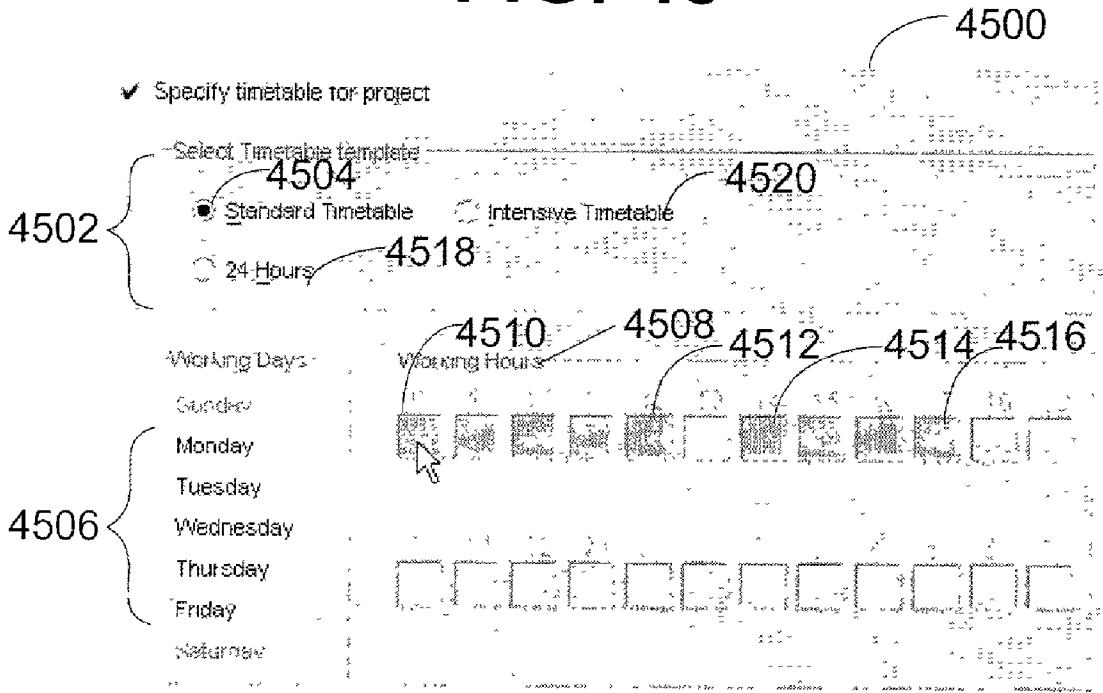


FIG. 46

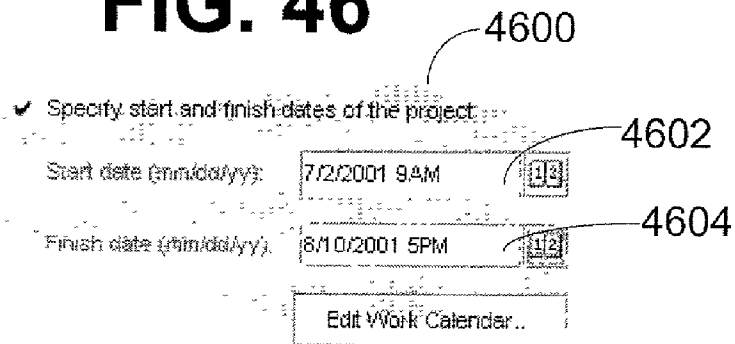


FIG. 47

4700



```

<?xml version="1.0" encoding="UTF-8" ?>
<process group="" name="Serial & Parallel">
  <description />
  <role name="Worker" />
  <activity id="10" name="Serial 1" responsiblerole="Worker">
    <description />
    <defduration units="hours" value="9" /> 4704
    <predecessors_id id="11" />
    <successors_id id="1522" />
  </activity>
  <activity id="1523" name="Parallel 1" responsiblerole="Worker">
    <description />
    <defduration units="hours" value="24" /> 4710
    <predecessors_id id="1522" />
    <successors_id id="1525" />
  </activity>
  <activity id="1524" name="Parallel 2" responsiblerole="Worker">
    <description />
    <defduration units="hours" value="24" /> 4712
    <predecessors_id id="1522" />
    <successors_id id="1525" />
  </activity>
  <activity id="1526" name="Serial 2" responsiblerole="Worker">
    <description />
    <defduration units="hours" value="24" /> 4716
    <predecessors_id id="1525" />
    <successors_id id="12" />
  </activity>
  <syncbar id="11">
    <description />
    <startBar>T</startBar>
    <successors_id id="10" />
  </syncbar>
  <syncbar id="12">
    <description />
    <endBar>T</endBar>
    <predecessors_id id="1526" />
  </syncbar>
  <syncbar id="1522" name="SyncBarH1">
    <description />
    <predecessors_id id="10" />
    <successors_id id="1523" />
    <successors_id id="1524" />
  </syncbar>
  <syncbar id="1525" name="SyncBarH2">
    <description />
    <predecessors_id id="1523" />
    <predecessors_id id="1524" />
    <successors_id id="1526" />
  </syncbar>
</process>

```

4702

4706

4708

4714

FIG. 48

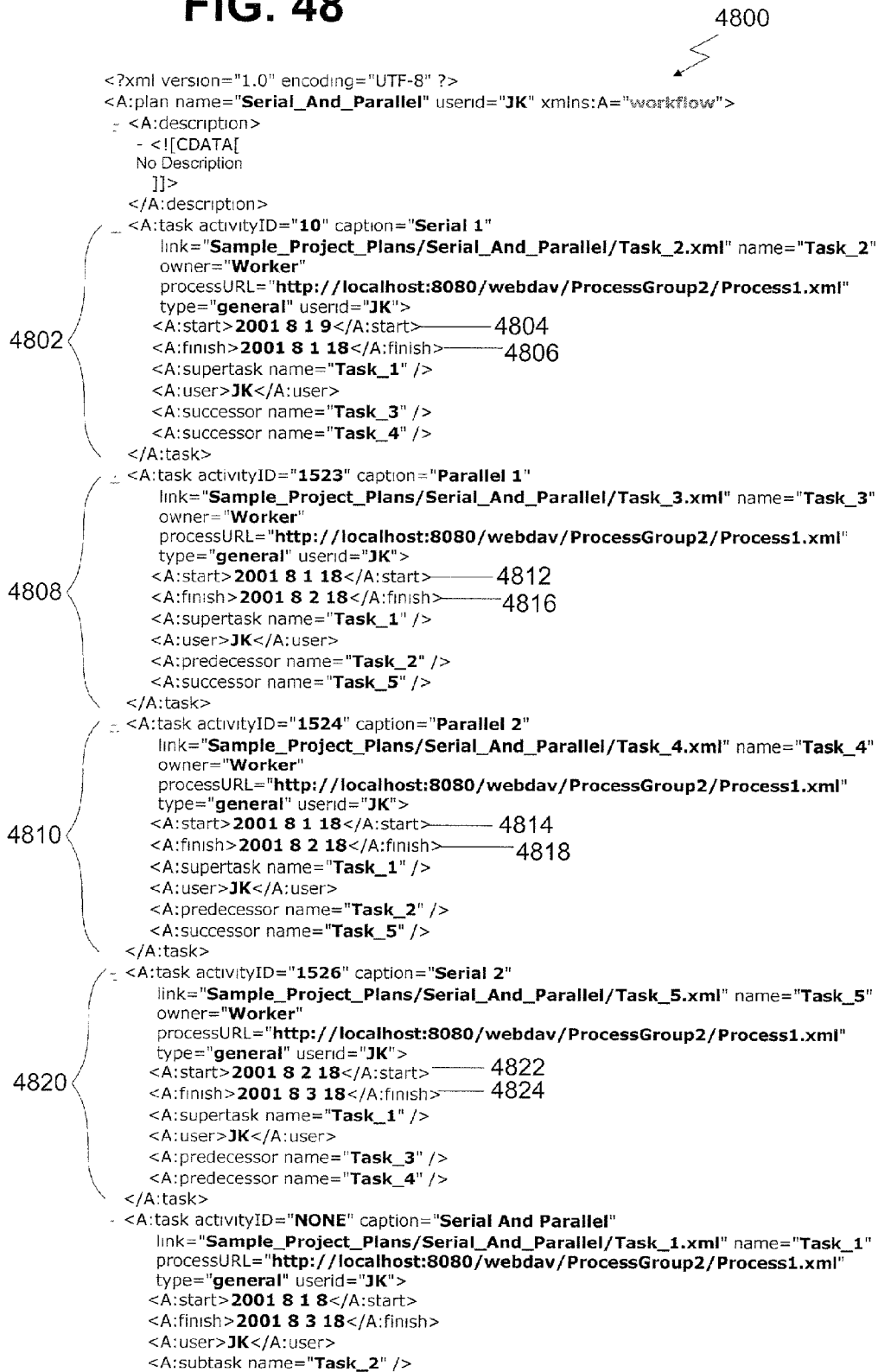


FIG. 49

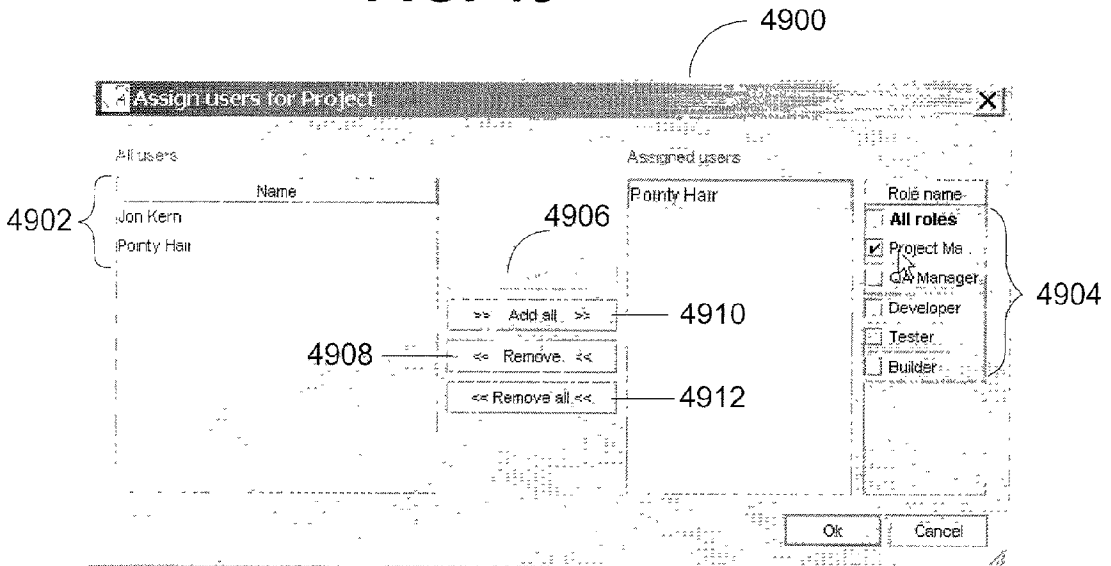


FIG. 50

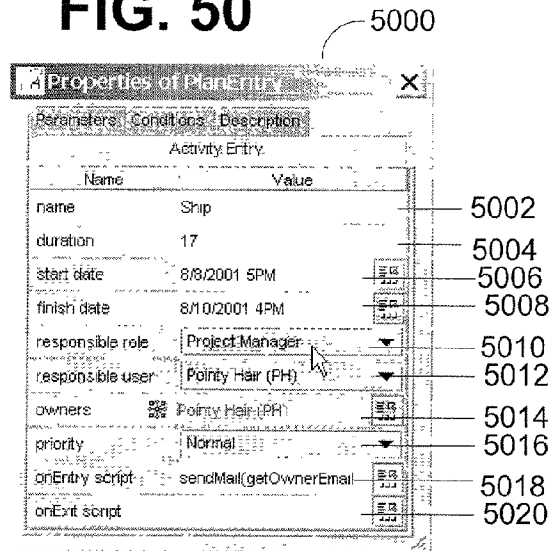


FIG. 51

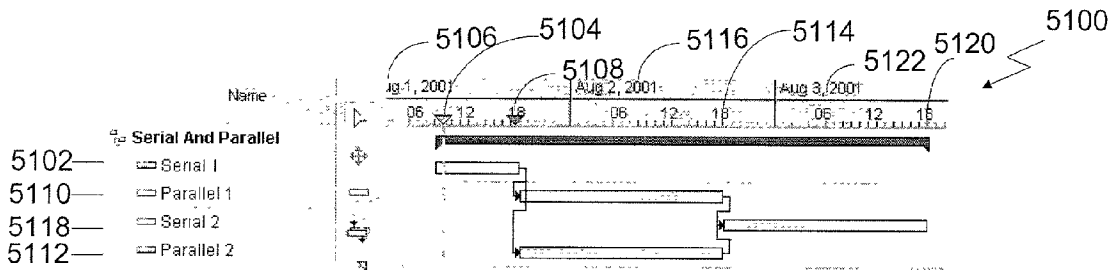


FIG. 52

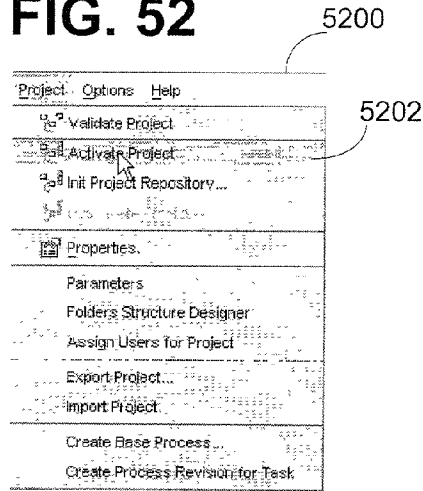


FIG. 53

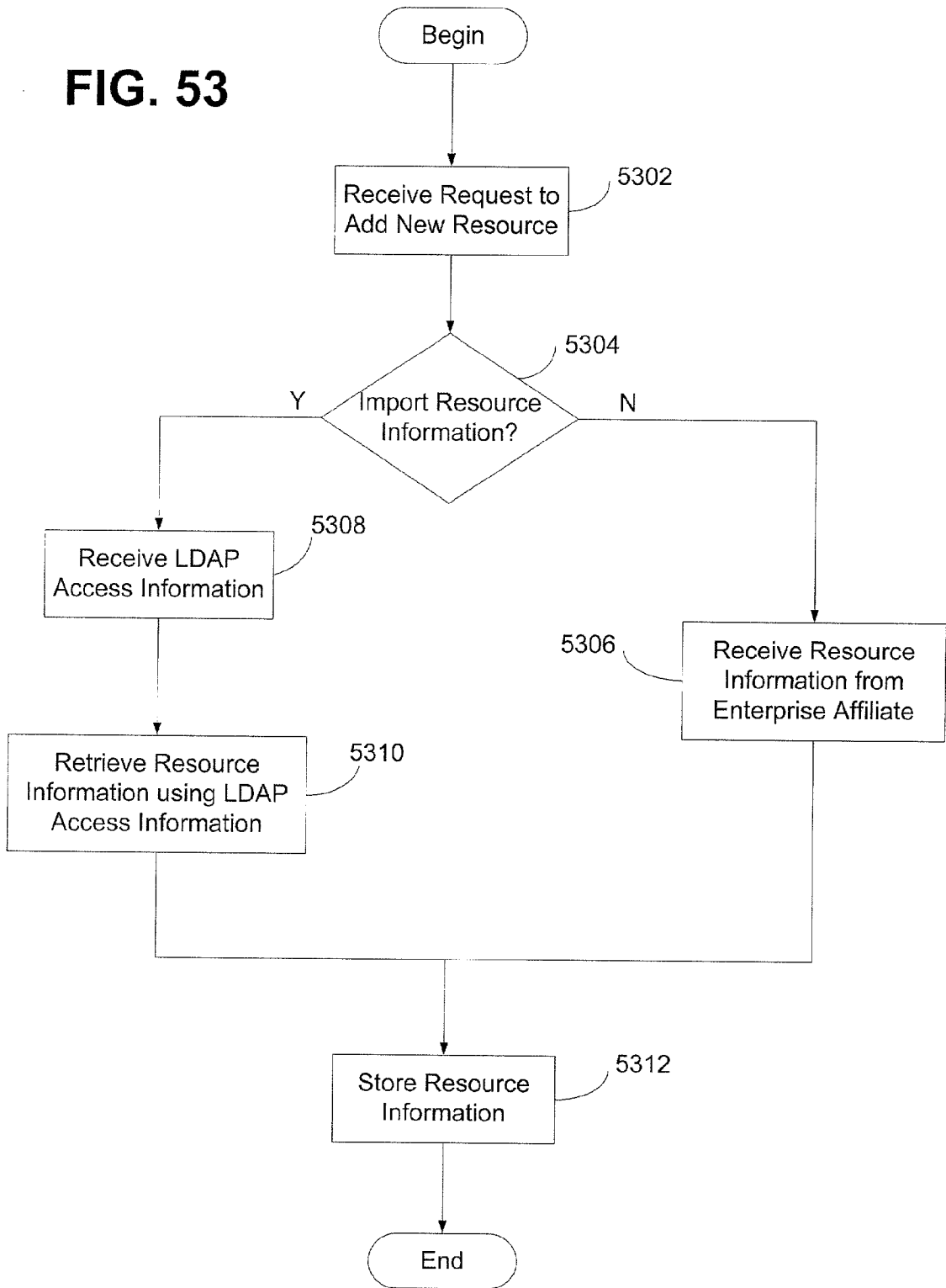
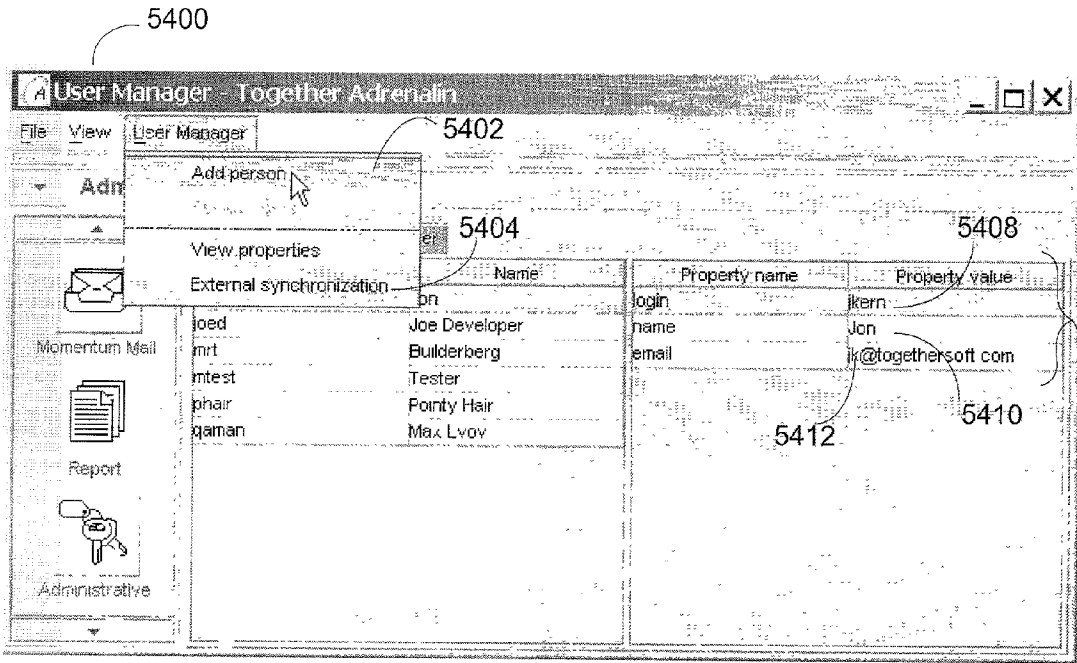


FIG. 54



5406

FIG. 55

5500

External synchronization X

New 389

5502 {

LDAP Server — 5504: frodo

LDAP Port — 5506: 389

People DN — 5508: OU=LOCAL USERS, DC=amazosoft, DC=com

Use password

Security DN — 5510:

Password — 5512:

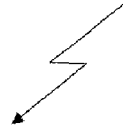
Login alias — 5514: mlogin

Add Remove Save Close Synchronize

The image shows a screenshot of a software dialog box titled "External synchronization". The dialog box has a title bar with a close button (X) on the right. Below the title bar, there is a text field containing "New 389". The main area of the dialog contains several labeled input fields. A bracket on the left side groups the first three fields: "LDAP Server" (5504) with the value "frodo", "LDAP Port" (5506) with the value "389", and "People DN" (5508) with the value "OU=LOCAL USERS, DC=amazosoft, DC=com". Below these is a checkbox labeled "Use password" which is currently unchecked. Further down are "Security DN" (5510) and "Password" (5512) fields, both of which are empty. The "Login alias" (5514) field contains the text "mlogin". At the bottom of the dialog, there is a row of five buttons: "Add", "Remove", "Save", "Close", and "Synchronize".

FIG. 56

5600



```
<?xml version="1.0" encoding="UTF-8" ?>
<users>
  <user>
    <id>PH</id>
    <name>Pointy Hair</name>
    <email>ph@company.com</email>
    <password>ZGV364==</password>
    url= "Software_Projects/Hello-World_App/Users_1.xml"
  </user>
  <user>
    <id>MQ</id>
    <name>Mister Quality</name>
    <email>mq@company.com</email>
    <password>ZGV636==</password>
    url= "Software_Projects/Hello-World_App/Users_1.xml"
  </user>
  <user>
    <id>MB</id>
    <name>Mister Build</name>
    <email>mb@company.com</email>
    <password>ZGV835==</password>
    url= "Software_Projects/Hello-World_App/Users_1.xml"
  </user>
  <user>
    <id>MT</id>
    <name>Mister Tee</name>
    <email>mt@company.com</email>
    <password>ZGV470==</password>
    url= "Software_Projects/Hello-World_App/Users_1.xml"
  </user>
</users>
```

5602 {

5604 {

5606 {

5608 {

5610

5612

FIG. 57A

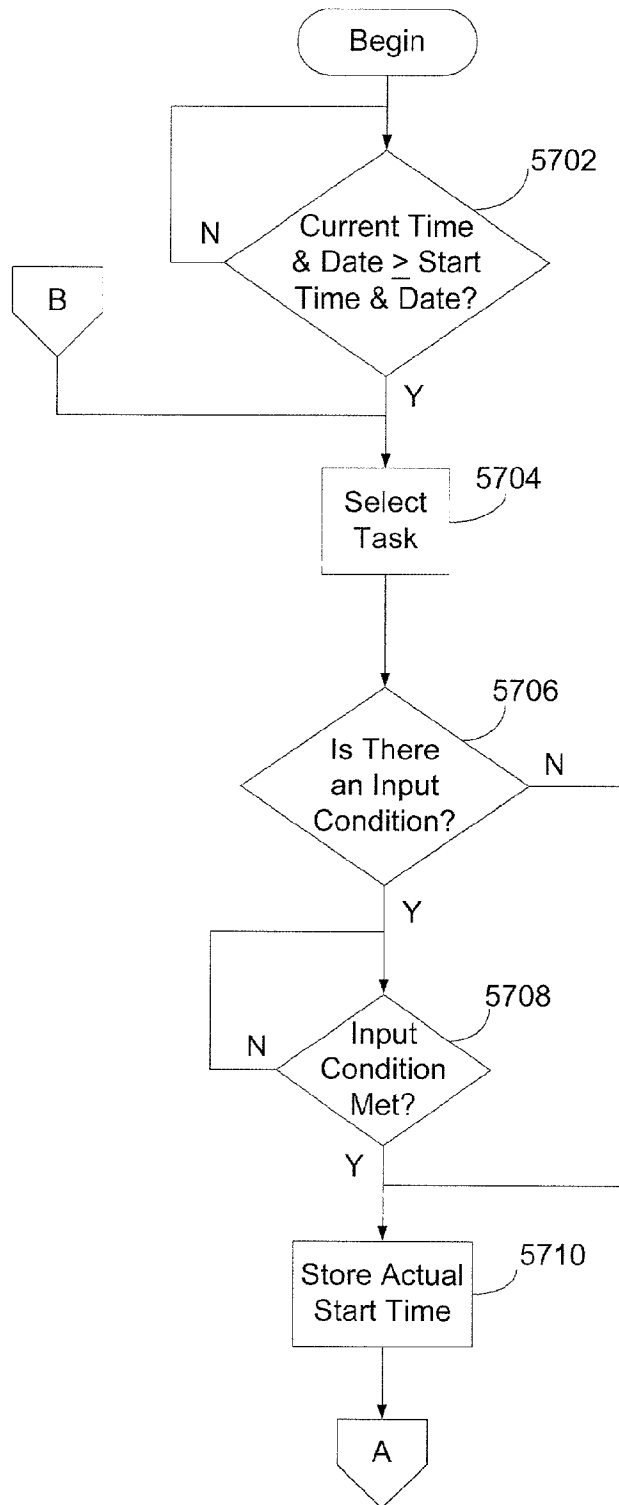


FIG. 57B

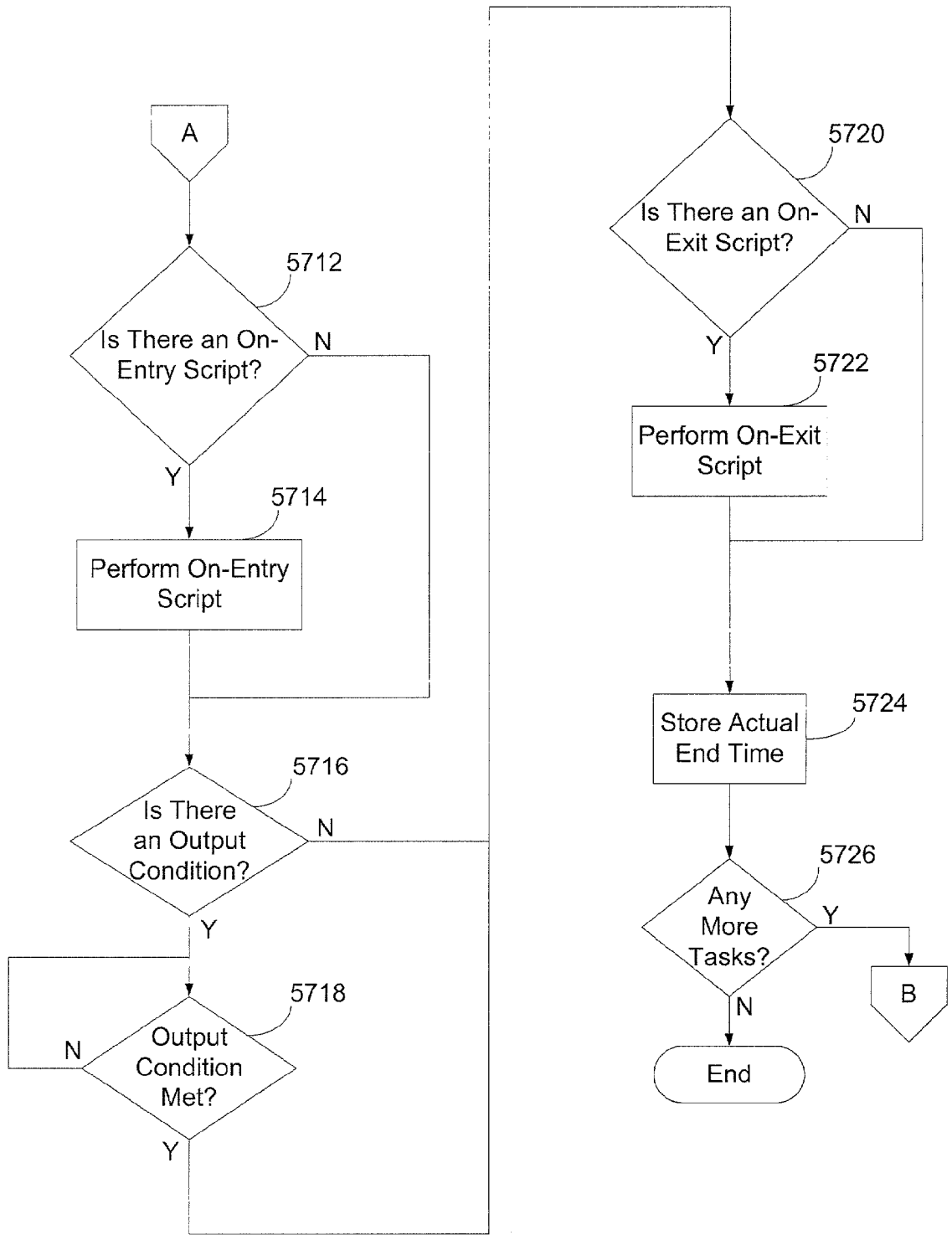


FIG. 58

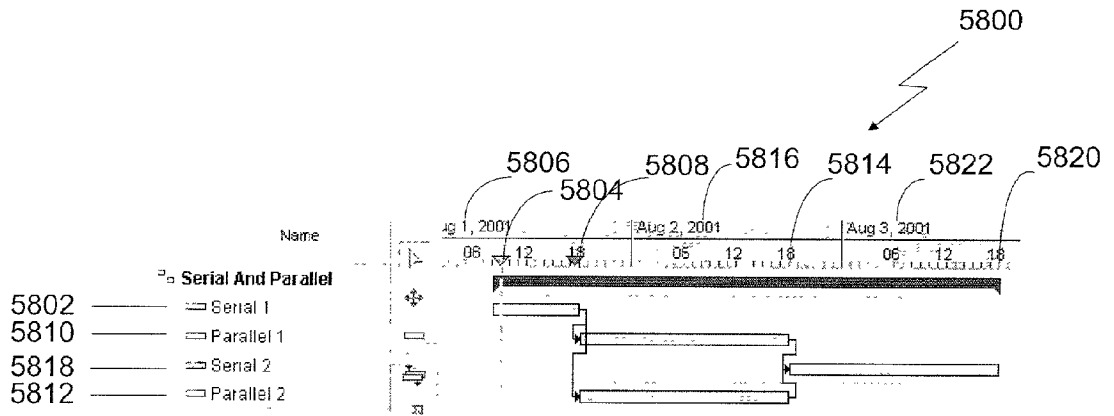


FIG. 59

5900



```

<?xml version="1.0" encoding="UTF-8" ?>
<A:plan name="Serial_And_Parallel" userid="JK" xmlns:A="workflow">
  <A:description>
    - <![CDATA[
      No Description
    ]]>
  </A:description>
  <A:task activityID="10" caption="Serial 1"
    link="Sample_Project_Plans/Serial_And_Parallel/Task_2.xml" name="Task_2"
    owner="Worker"
    processURL="http://localhost:8080/webdav/ProcessGroup2/Process1.xml"
    type="general" userid="JK">
    <A:start>2001 8 1 8</A:start>
    <A:finish>2001 8 1 18</A:finish>
    <A:supertask name="Task_1" />
    <A:user>JK</A:user>
    <A:successor name="Task_3" />
    <A:successor name="Task_4" />
  </A:task>
  <A:task activityID="1523" caption="Parallel 1"
    link="Sample_Project_Plans/Serial_And_Parallel/Task_3.xml" name="Task_3"
    owner="Worker"
    processURL="http://localhost:8080/webdav/ProcessGroup2/Process1.xml"
    type="general" userid="JK">
    <A:start>2001 8 1 18</A:start>
    <A:finish>2001 8 2 18</A:finish>
    <A:supertask name="Task_1" />
    <A:user>JK</A:user>
    <A:predecessor name="Task_2" />
    <A:successor name="Task_5" />
  </A:task>
  <A:task activityID="1524" caption="Parallel 2"
    link="Sample_Project_Plans/Serial_And_Parallel/Task_4.xml" name="Task_4"
    owner="Worker"
    processURL="http://localhost:8080/webdav/ProcessGroup2/Process1.xml"
    type="general" userid="JK">
    <A:start>2001 8 1 18</A:start>
    <A:finish>2001 8 2 18</A:finish>
    <A:supertask name="Task_1" />
    <A:user>JK</A:user>
    <A:predecessor name="Task_2" />
    <A:successor name="Task_5" />
  </A:task>
  <A:task activityID="1526" caption="Serial 2"
    link="Sample_Project_Plans/Serial_And_Parallel/Task_5.xml" name="Task_5"
    owner="Worker"
    processURL="http://localhost:8080/webdav/ProcessGroup2/Process1.xml"
    type="general" userid="JK">
    <A:start>2001 8 2 18</A:start>
    <A:finish>2001 8 3 18</A:finish>
    <A:supertask name="Task_1" />
    <A:user>JK</A:user>
    <A:predecessor name="Task_3" />
    <A:predecessor name="Task_4" />
  </A:task>
  <A:task activityID="NONE" caption="Serial And Parallel"
    link="Sample_Project_Plans/Serial_And_Parallel/Task_1.xml" name="Task_1"
    processURL="http://localhost:8080/webdav/ProcessGroup2/Process1.xml"
    type="general" userid="JK">
    <A:start>2001 8 1 8</A:start>
    <A:finish>2001 8 3 18</A:finish>
    <A:user>JK</A:user>
    <A:subtask name="Task_2" />
  </A:task>

```

FIG. 60

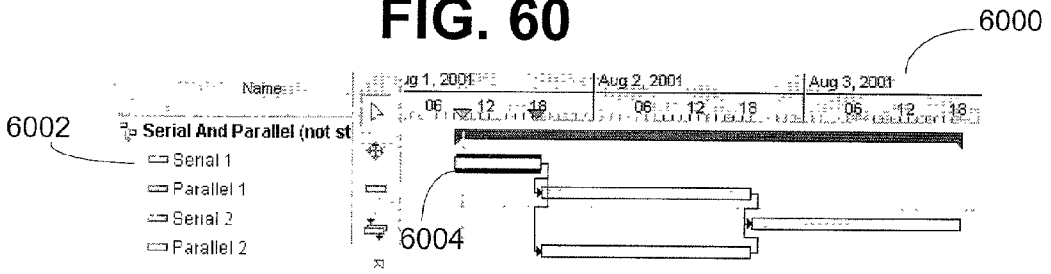


FIG. 61

```

<?xml version="1.0" encoding="UTF-8" ?>
<A:FilePropertyMap xmlns:A="adrenalin:">
  <A:factstart xmlns:A="workflow">2001 8 1 9</A:factstart>
  <A:factfinish xmlns:A="workflow">2001 8 1 14</A:factfinish>
  <A:ProjectTask xmlns:A="Serial 1" />
</A:FilePropertyMap>
    
```

FIG. 61 shows XML code for a file property map. The code defines a workflow with a factstart of 2001 8 1 9 and a factfinish of 2001 8 1 14. It includes a project task for 'Serial 1'. Callouts 6100, 6104, 6102, and 6106 point to various parts of the XML structure.

FIG. 62

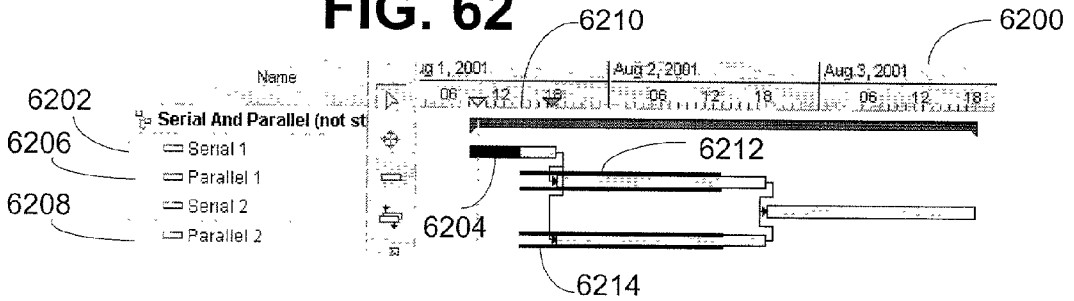


FIG. 63

```

<?xml version="1.0" encoding="UTF-8" ?>
<A:FilePropertyMap xmlns:A="adrenalin:">
  <A:factstart xmlns:A="workflow">2001 8 1 14</A:factstart>
  <A:factfinish xmlns:A="workflow">2001 8 2 0</A:factfinish>
  <A:ProjectTask xmlns:A="Parallel 1" />
  <A:ProjectTask xmlns:A="Parallel 2" />
</A:FilePropertyMap>
    
```

FIG. 63 shows XML code for a file property map. The code defines a workflow with a factstart of 2001 8 1 14 and a factfinish of 2001 8 2 0. It includes two project tasks for 'Parallel 1' and 'Parallel 2'. Callouts 6300, 6306, 6308, 6302, and 6304 point to various parts of the XML structure.

FIG. 64

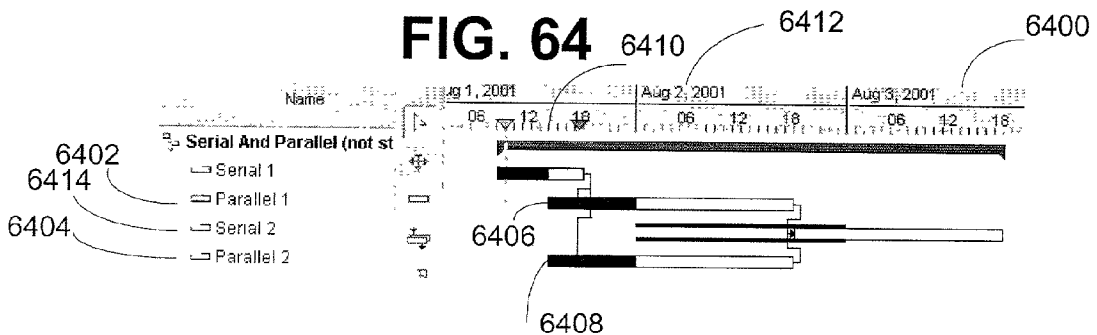


FIG. 65

```

        6500
    <?xml version="1.0" encoding="UTF-8" ?>
    - <A:FilePropertyMap xmlns:A="adrenalin:">
      <A:factstart xmlns:A="workflow">2001 8 2 0</A:factstart>
      <A:factfinish xmlns:A="workflow">2001 8 2 12</A:factfinish>
      <A:ProjectTask xmlns:A="Serial 2" />
    </A:FilePropertyMap>
    
```

6504

6506

6502

FIG. 66

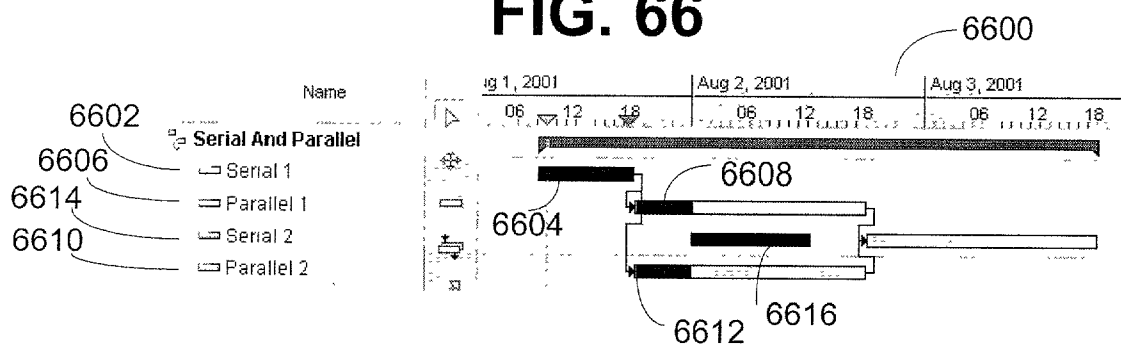


FIG. 67

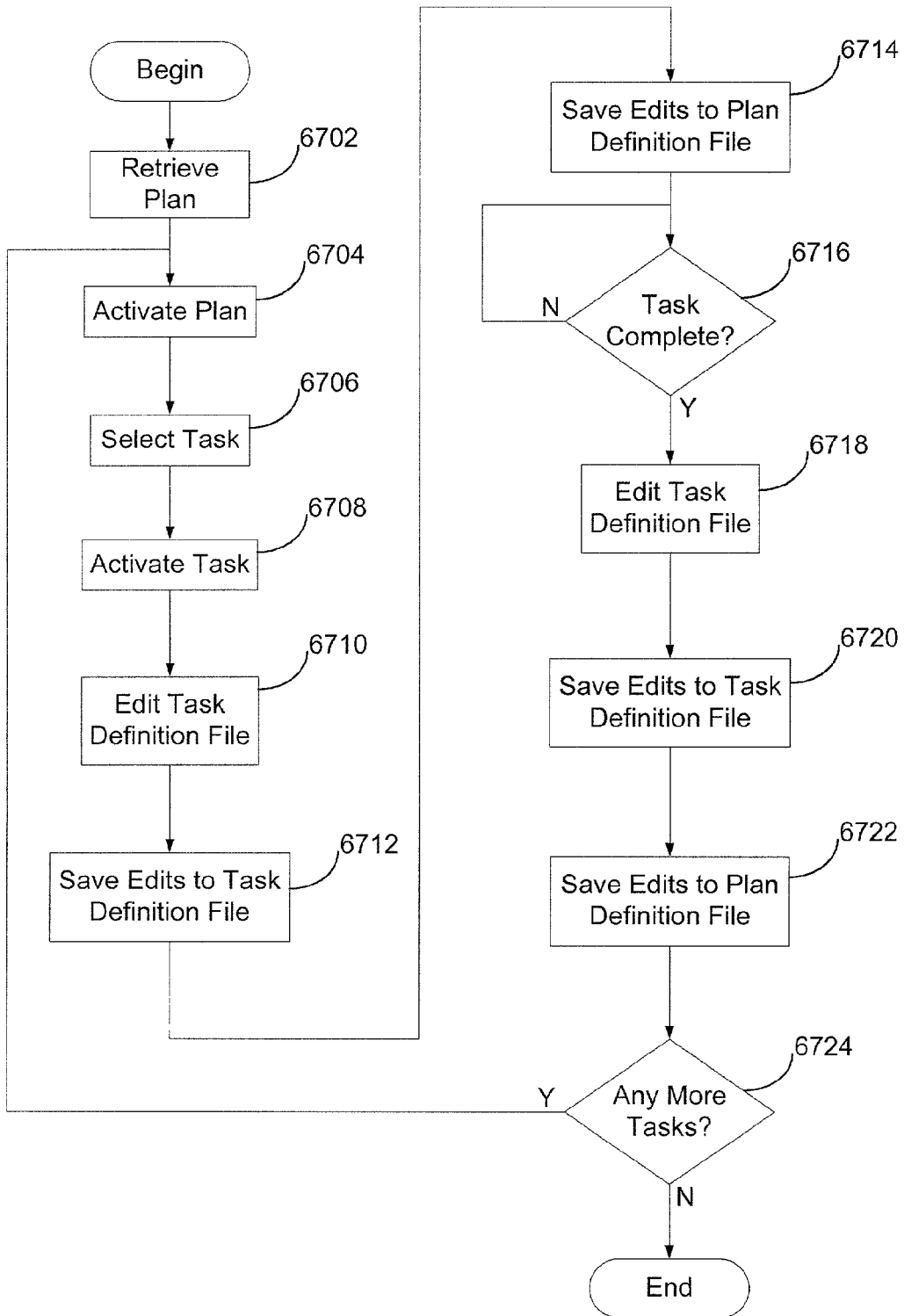


FIG. 68

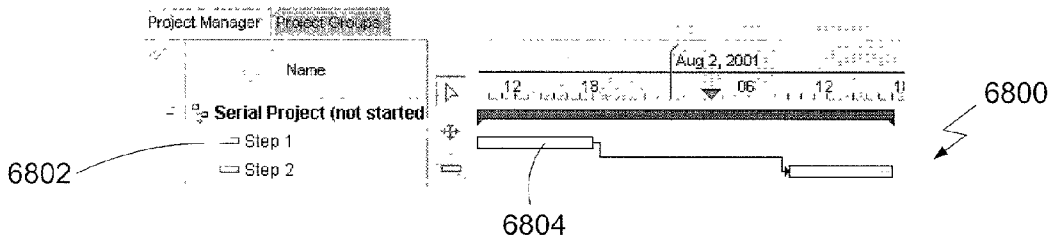


FIG. 69

```

- <A:task activityID="10" caption="Step 1"
  link="Sample_Project_Plans/Serial_Project/Task_2.xml" name="Task_2"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 1 9</A:start>
  <A:finish>2001 8 1 18</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:successor name="Task_3" />
  <state>UNEXECUTED</state>
</A:task>
    
```

6900

6902

FIG. 70

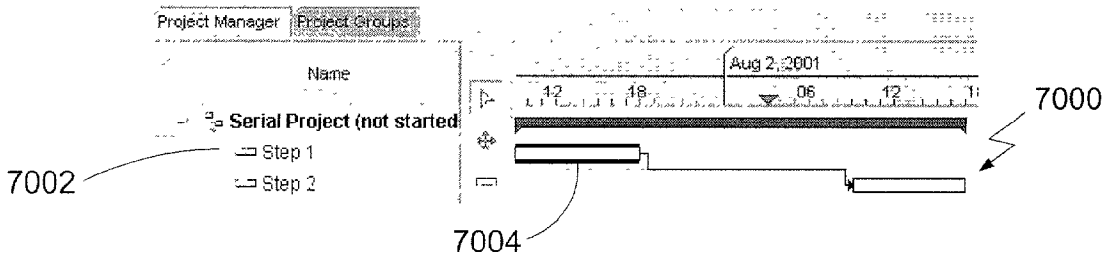


FIG. 71

```

<A:task activityID="10" caption="Step 1"
  link="Sample_Project_Plans/Serial_Project/Task_2.xml" name="Task_2"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 1 9</A:start>
  <A:finish>2001 8 1 18</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:successor name="Task_3" />
  <state>EXECUTING</state>
</A:task>
    
```

7100

7102

FIG. 72

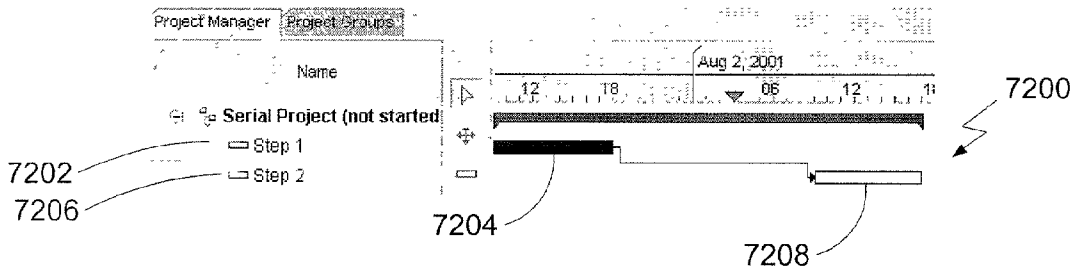


FIG. 73

```

- <A:task activityID="10" caption="Step 1"
  link="Sample_Project_Plans/Serial_Project/Task_2.xml" name="Task_2"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 1 9</A:start>
  <A:finish>2001 8 1 18</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:successor name="Task_3" />
  <state>EXECUTED</state>
</A:task>
    
```

FIG. 74

```

<A:task activityID="1527" caption="Step 2"
  link="Sample_Project_Plans/Serial_Project/Task_3.xml" name="Task_3"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 2 9</A:start>
  <A:finish>2001 8 2 17</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:predecessor name="Task_2" />
  <state>UNEXECUTED</state>
</A:task>
    
```

FIG. 75

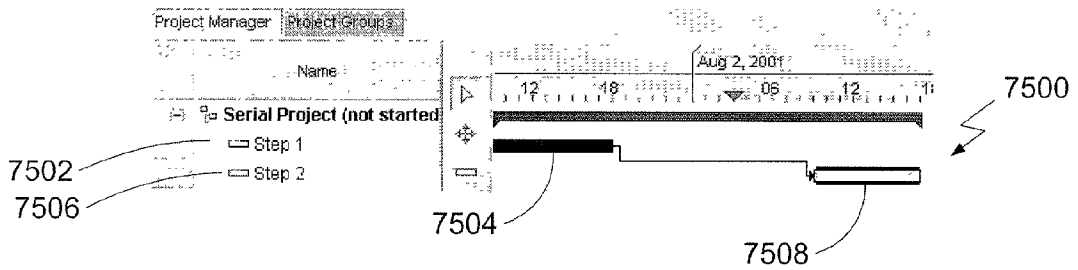


FIG. 76

```

- <A:task activityID="1527" caption="Step 2"
  link="Sample_Project_Plans/Serial_Project/Task_3.xml" name="Task_3"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 2 9</A:start>
  <A:finish>2001 8 2 17</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:predecessor name="Task_2" />
  <state>EXECUTING</state>
</A:task>
    
```

7600

7602

FIG. 77

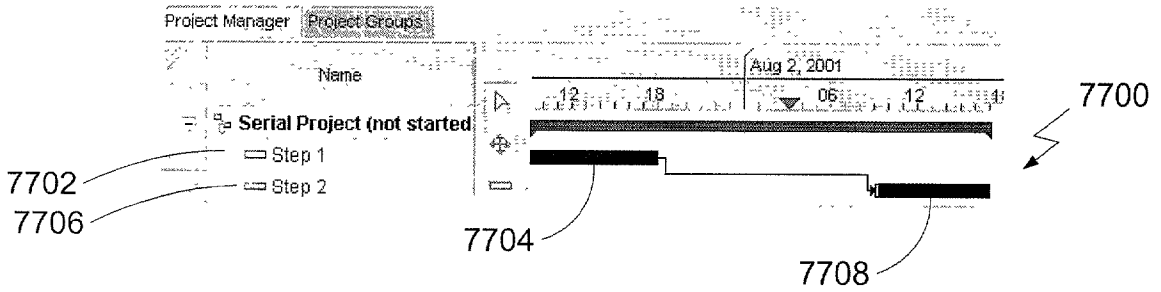


FIG. 78

```

- <A:task activityID="1527" caption="Step 2"
  link="Sample_Project_Plans/Serial_Project/Task_3.xml" name="Task_3"
  owner="Worker"
  processURL="http://localhost:8080/webdav/ProcessGroup2/Process2.xml"
  type="general" userid="JK">
  <A:start>2001 8 2 9</A:start>
  <A:finish>2001 8 2 17</A:finish>
  <A:supertask name="Task_1" />
  <A:user>JK</A:user>
  <A:predecessor name="Task_2" />
  <state>EXECUTED</state>
</A:task>
    
```

7800

7802

FIG. 79

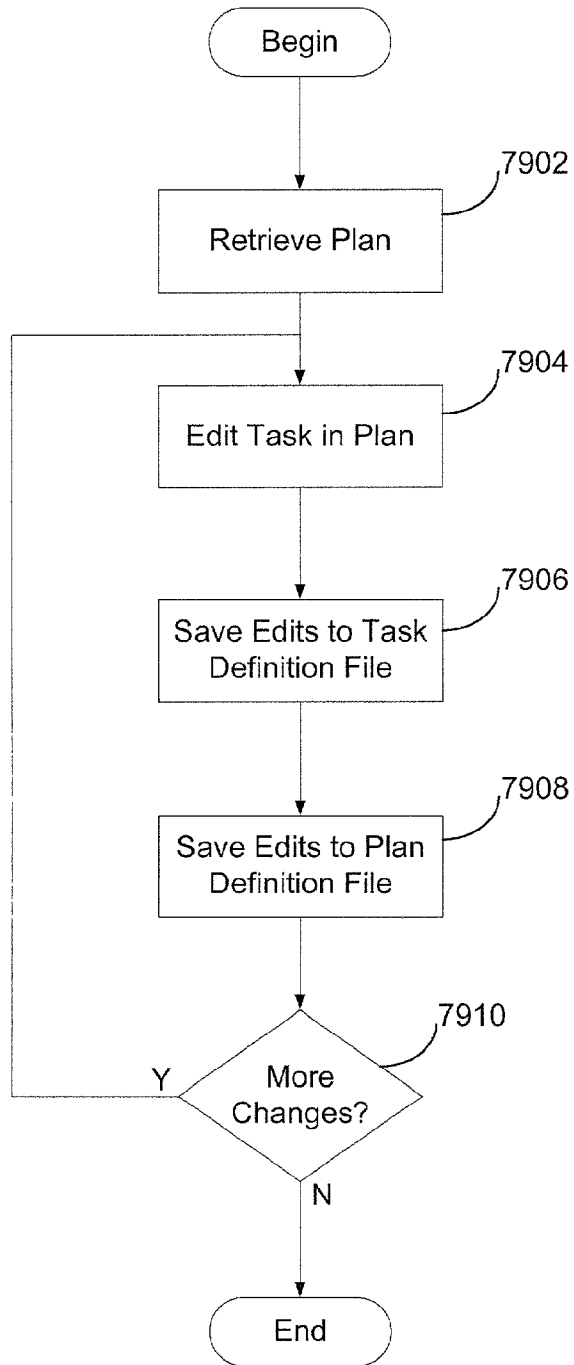


FIG. 80A

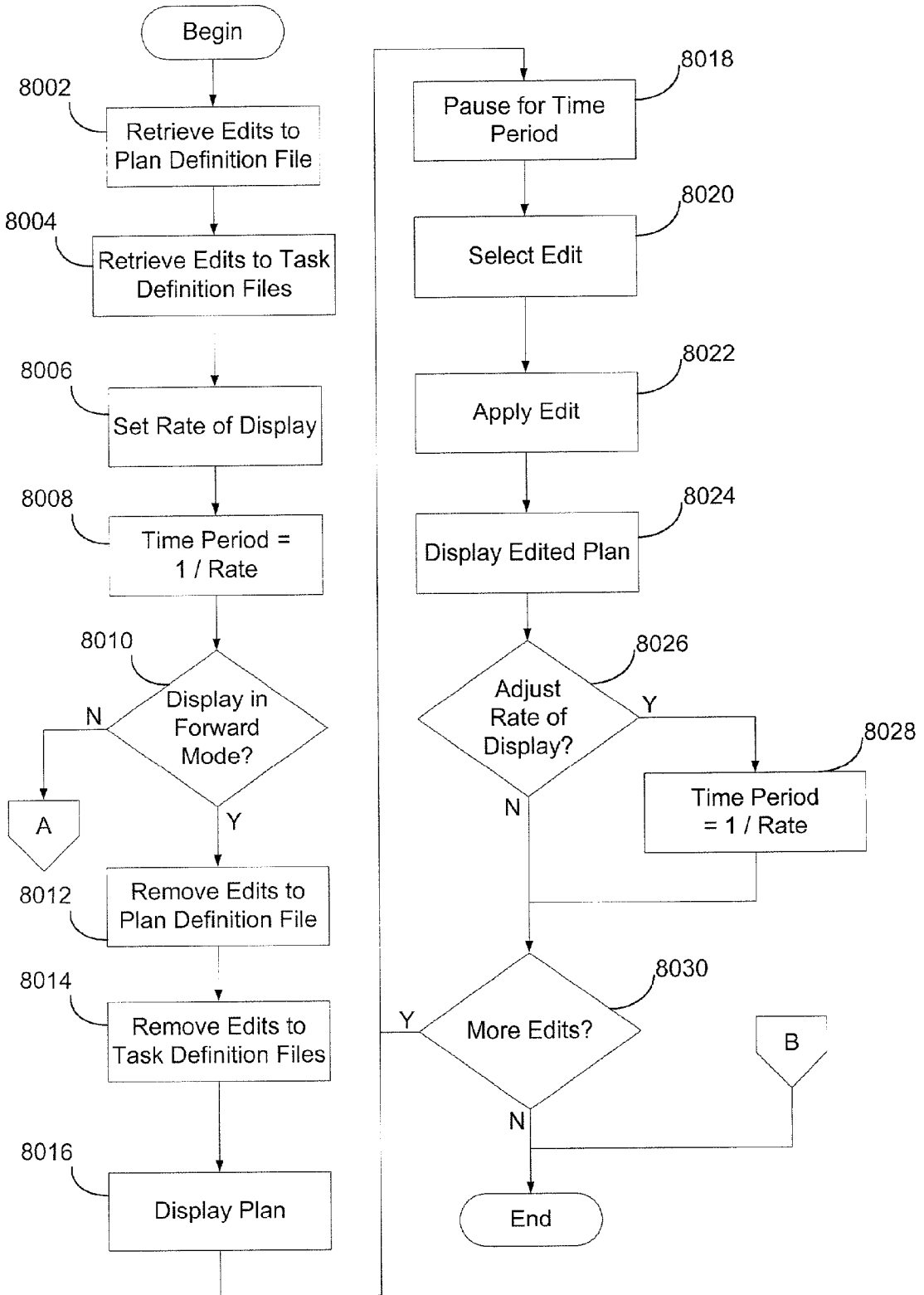


FIG. 80B

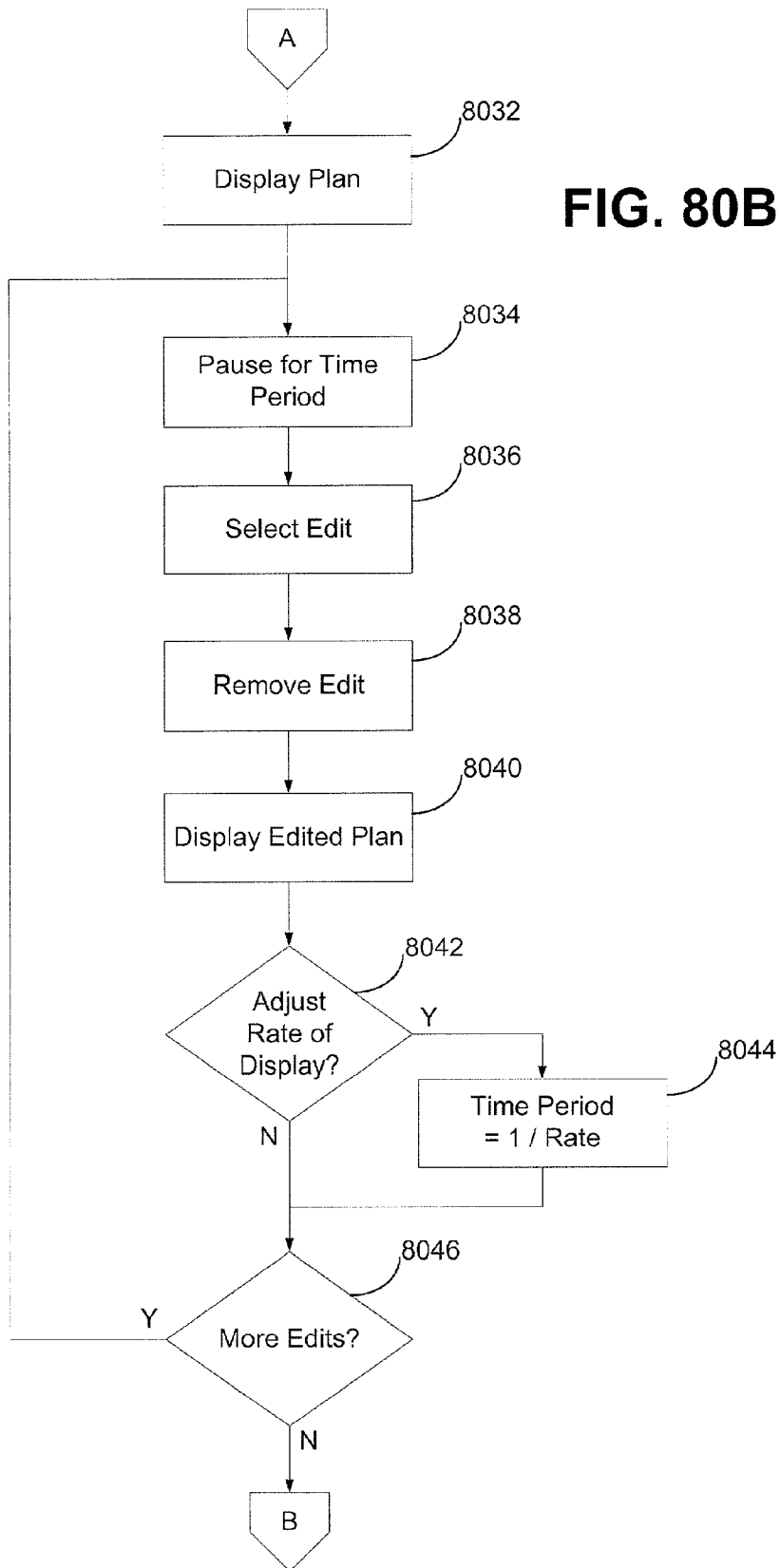


FIG. 81

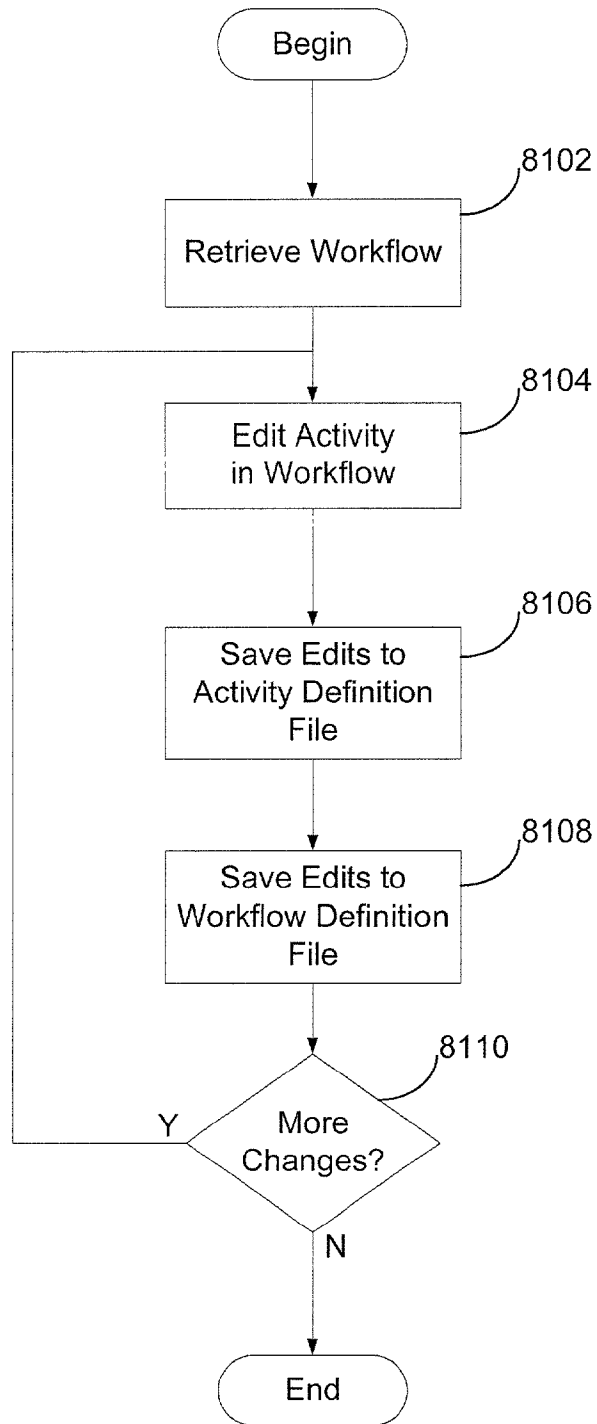


FIG. 82

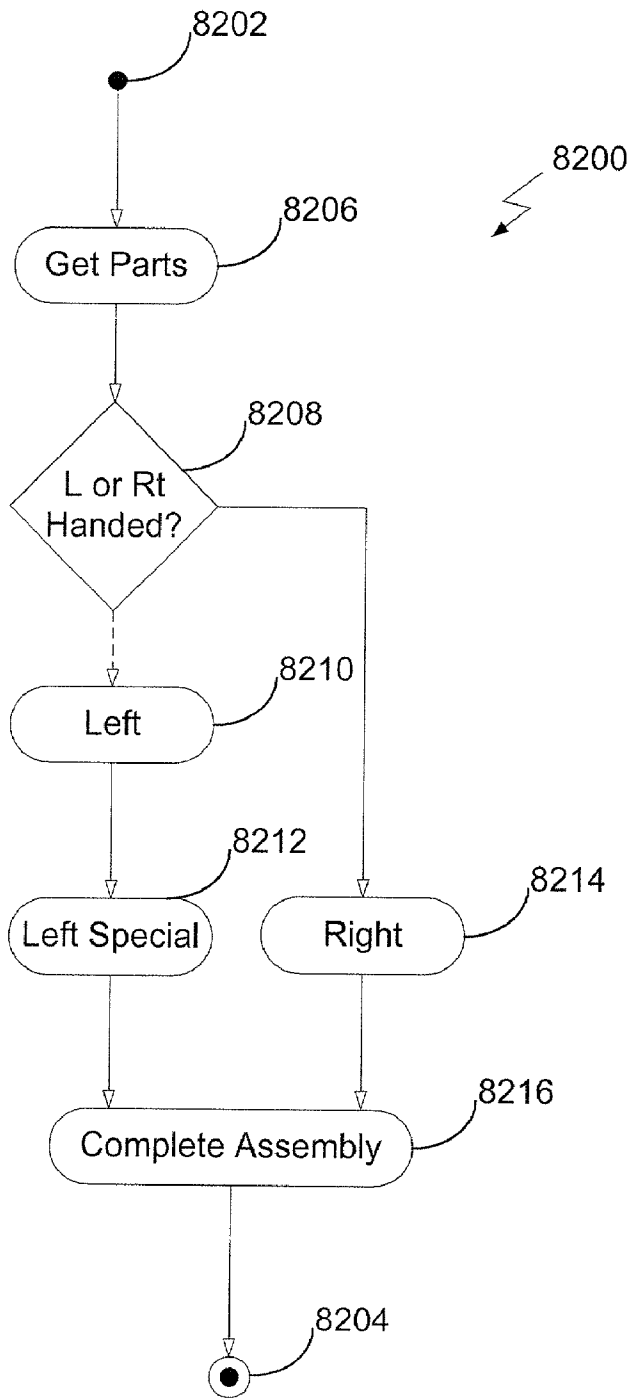


FIG. 83

8302

```
- <activity id="10" name="Get Parts" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="11" />  
  <successors_id id="1522" />  
  <out_artifacts id="1527" />  
</activity>
```

8304

8300

FIG. 84

```
- <activity id="1525" name="Right" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1522" />  
  <successors_id id="1526" />  
  <out_artifacts id="1533" />  
</activity>
```

8400

FIG. 85

8502

```
- <activity id="1526" name="Complete Assembly" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1524" />  
  <predecessors_id id="1525" />  
  <successors_id id="12" />  
  <out_artifacts id="1535" />  
</activity>
```

8504

8500

FIG. 86

```
- <activity id="1523" name="Left" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1522" />  
  <successors_id id="1524" />  
  <out_artifacts id="1529" />  
</activity>
```

8602

8600

FIG. 87

8702

```
- <activity id="1524" name="Left Special" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1523" />  
  <successors_id id="1526" />  
  <out_artifacts id="1531" />  
</activity>
```

8700

FIG. 88

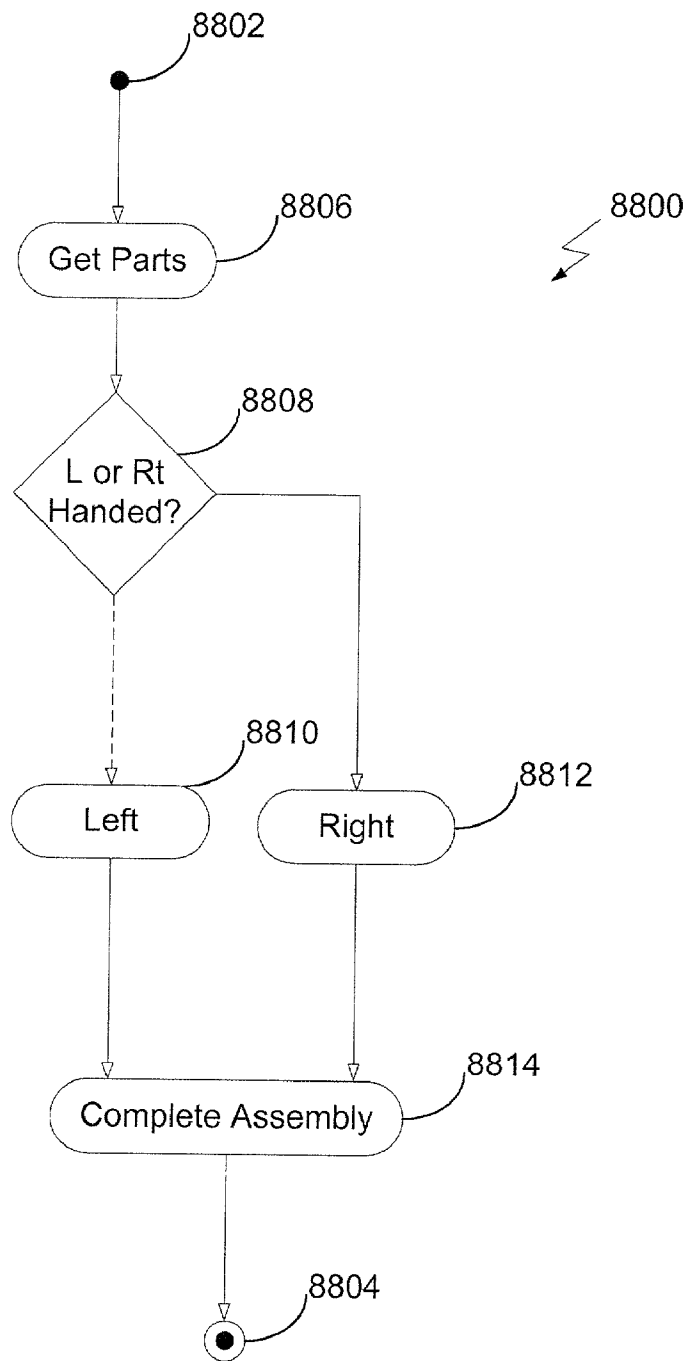


FIG. 89

8900

```
- <activity id="1523" name="Left" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1522" />  
  <successors_id id="1526" />  
  <out_artifacts id="1529" />  
</activity>
```

8902

FIG. 90

9000

```
- <activity id="1526" name="Complete Assembly" responsiblerole="Assembler">  
  <description />  
  <defduration units="units" value="1" />  
  <predecessors_id id="1523" />  
  <predecessors_id id="1525" />  
  <successors_id id="12" />  
  <out_artifacts id="1535" />  
</activity>
```

9002

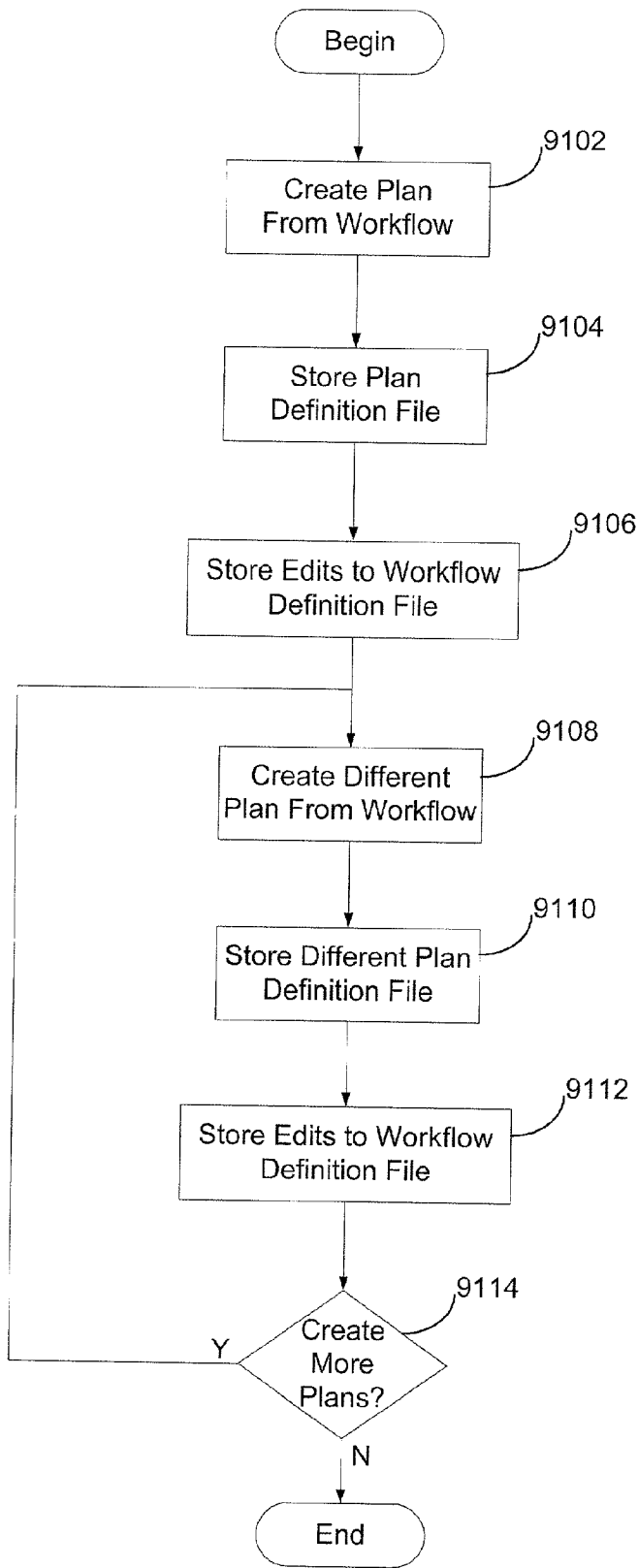


FIG. 91

FIG. 92A

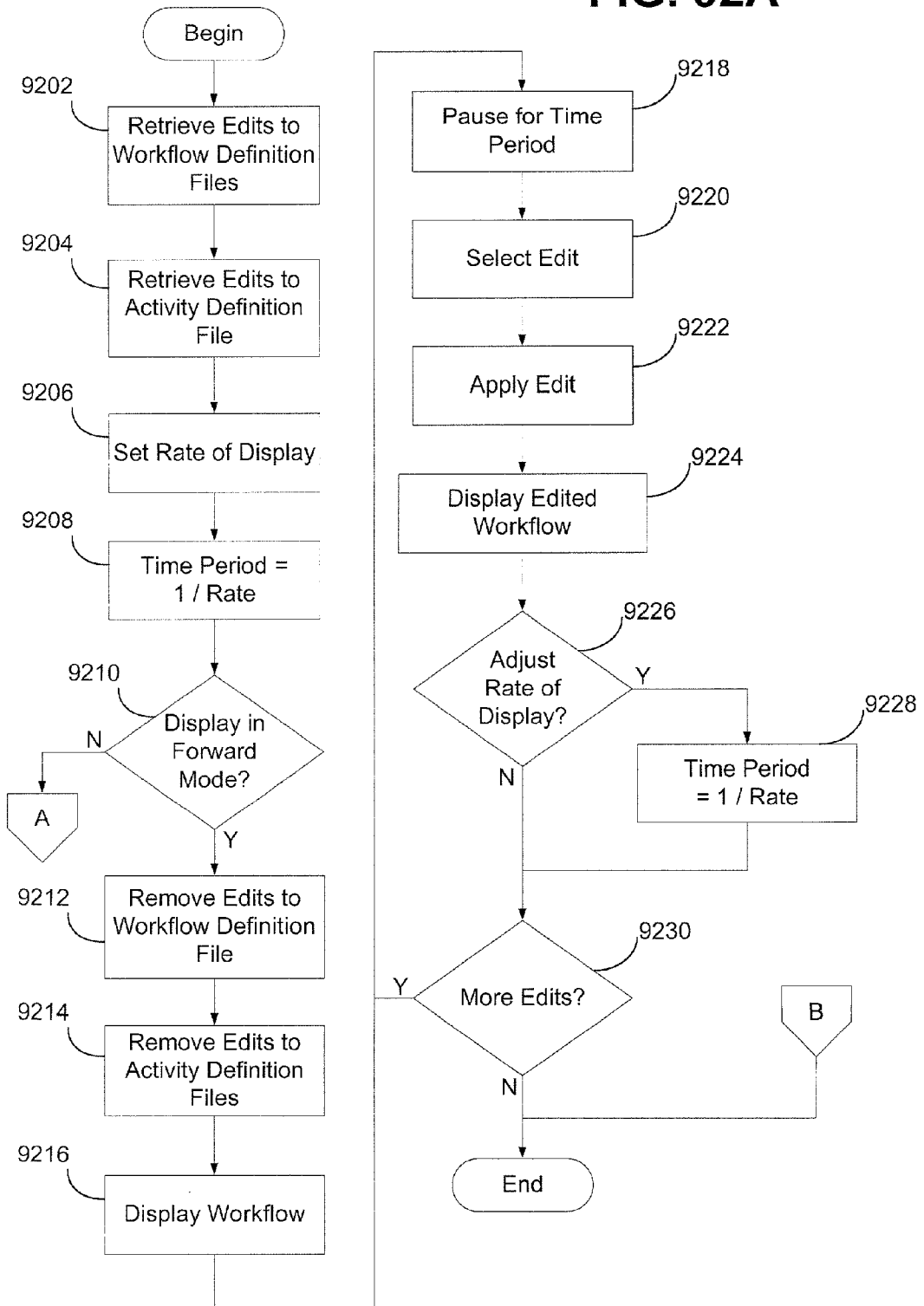
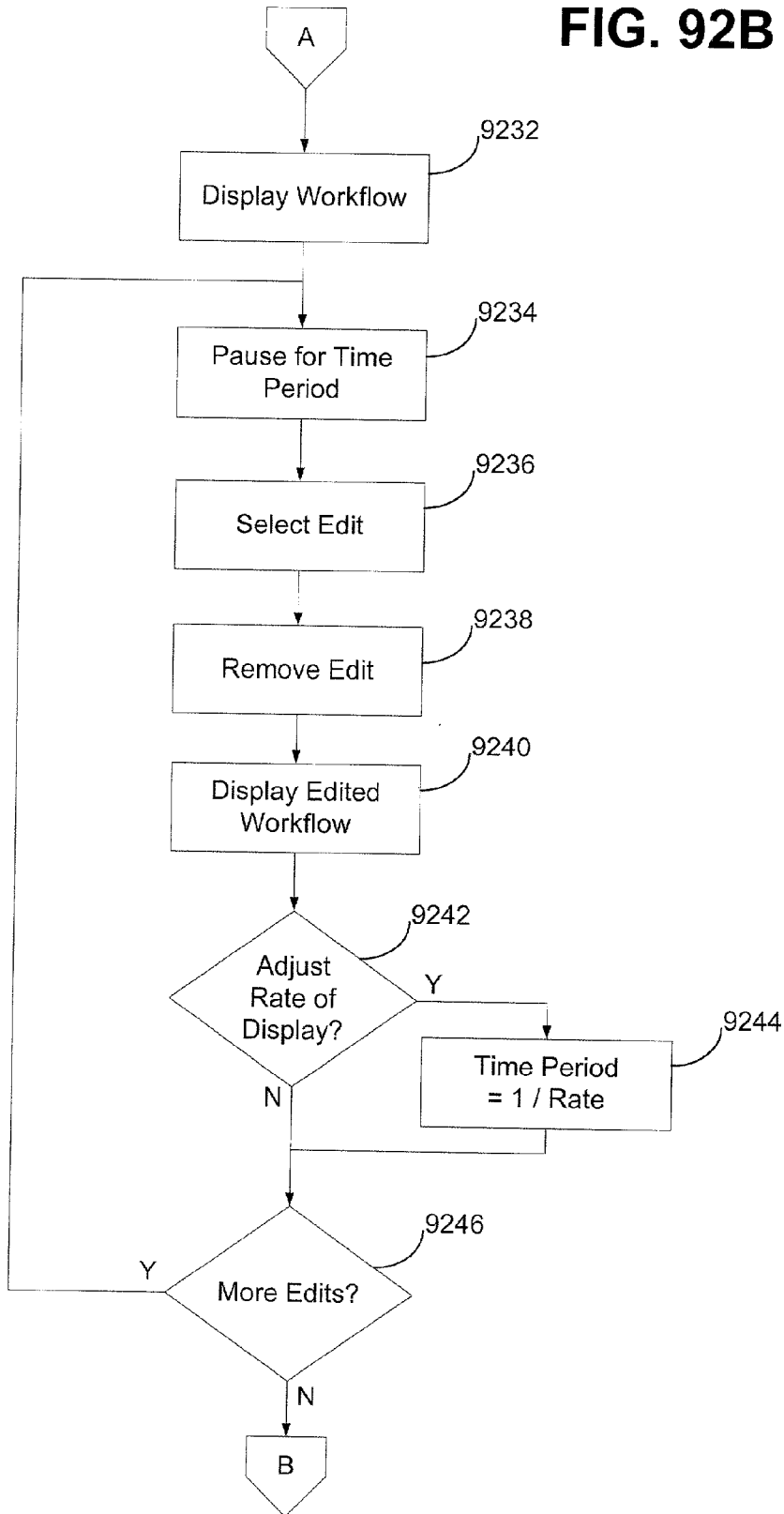


FIG. 92B



METHODS AND SYSTEMS FOR ANIMATING A WORKFLOW AND A PROJECT PLAN

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of the filing date of U.S. Provisional Application No. 60/230,054, entitled "Development Tool for Modeling Workflow," filed on Sep. 1, 2000, and U.S. Provisional Application No. 60/296,707, entitled "Improved Development Tool For Modeling Workflow," filed on Jun. 7, 2001, both of which are incorporated herein by reference.

[0002] The following identified U.S. patent applications are also relied upon and are incorporated by reference in this application:

[0003] U.S. patent application Ser. No. _____, entitled "Methods and Systems for Integrating Process Modeling and Project Planning," bearing attorney docket no. TS1000, and filed on the same date herewith;

[0004] U.S. patent application Ser. No. _____, entitled "Methods and Systems for Improving a Workflow Based on Data Mined from Plans Created from the Workflow," bearing attorney docket no. TS1001, and filed on the same date herewith; and

[0005] U.S. patent application Ser. No. _____, entitled "Methods and Systems for Optimizing Resource Allocation Based on Data Mined from Plans Created from a Workflow," bearing attorney docket no. TS1006, and filed on the same date herewith.

FIELD OF THE INVENTION

[0006] The present invention relates to a method and system for integrating a business process or workflow with a project plan. More particularly, the invention relates to a method and system for creating and activating a project plan, and animating the activation of the project plan. The method and system may also track changes made to the workflow or project plan, and animate the corresponding changes to the workflow or project plan. Additionally, the invention relates to a method and system for creating different project plans from one workflow, and animating the creation of the different project plans.

BACKGROUND OF THE INVENTION

[0007] To become more efficient and competitive, businesses and industries have striven to capture and streamline the business processes or workflows they use to operate and manage their respective enterprises. In general, a workflow is a model of a process. More specifically, a workflow can be viewed as a structured set of activities designed to produce a specific output for a particular customer (internal or external to an enterprise) or market. Although conventional software tools define the steps performed by the workflow, conventional tools do not schedule the resources (e.g., the people, equipment, or software technologies) responsible for completing each activity. Conventional tools also do not prepare a timeline identifying the beginning or end of each activity. Thus, conventional tools do not prepare a schedule for completing the workflow.

[0008] Businesses and industries also use other conventional software tools, such as Microsoft Project™, to build

and manage a project plan for their respective enterprises. A plan represents an instance of the workflow. More specifically, a plan can be viewed as a working schedule for a project to produce a product or artifact, such as a computer, bicycle, or software build, for the respective enterprise. These other conventional software tools typically display the working schedule in the form of an interactive Gantt chart, i.e., a chart to which the user can make updates. A Gantt chart is the linear, time-based representation of a project schedule, usually laid out on a horizontal plane where the times/dates increase to the right. These Gantt charts show the temporal relationships between the different tasks in a project, where the tasks are arranged along the vertical axis. Gantt charts are typically used to lay out an initial plan/timeline for the project, and then to track the actual progress of a project from start to finish. The modern software-based Gantt chart also identifies the resource(s) responsible for completing each task of the plan, the dependencies between the tasks, and, once the project has begun, the status of each task.

[0009] The conventional tools that support the building and managing of a project plan, however, do not provide direct links between projects and the workflows or business processes that the enterprise has defined and seeks to implement to gain business advantage and economies of efficiencies. Likewise, the conventional tools that enterprises use to define and manage workflows are not linked to project plans. Because both workflows and project plans do not exist on the same tool, workflows and project plans cannot be integrated or synchronized to keep the workflows and project plans "in step" with each other. Thus, there is a need in the art for a tool that avoids the limitations of these conventional software tools.

SUMMARY OF THE INVENTION

[0010] Methods and systems consistent with the present invention provide a workflow modeling and project planning integration tool that overcomes the limitations of conventional tools. Contrary to conventional tools that do not allow a user to integrate a business process or workflow with a project plan, the integration tool, in accordance with methods and systems consistent with the present invention, allows a user to model a business process or workflow, to create and activate or start a project plan based on the workflow, to manage the execution of the activated plan, and to track the progress of the activated project plan. In addition, the tool may include a Web-based "Distributed Authoring and Versioning" server that operates as a virtual file system to allow more than one user to view the same workflow or project plan, to provide persistent storage, to monitor the progress of an activated project plan, to simultaneously create plans from the same workflow, and to have essentially unlimited access to the power of the tool through the ubiquity of the Internet. "Versioning is a term well-known in the art for capturing the state of an entity at given points in time.

[0011] Methods and systems consistent with the present invention allow a user to animate different versions of a plan or workflow. Each version reflects an instance in an edit history, i.e., reflects the changes made to the plan or workflow. Additionally, methods and systems consistent with the present invention allow a user to view the various plans created from a given workflow over time. Methods and

systems consistent with the present invention also allow a user to review the steps performed during the activation of a plan. The user may adjust the rate at which the animation is displayed. The animation may also be viewed in reverse order.

[0012] In accordance with methods consistent with the present invention, a method is provided in a data processing system. The data processing system comprises versions of a plan, and each version reflects an instance in an edit history. The method comprises the steps of storing indications of the versions of the plan, and displaying the versions of the plan in a sequential manner to simulate animation of the edit history.

[0013] In accordance with methods consistent with the present invention, a method is provided in a data processing system. The data processing system comprises versions of a workflow, and each version reflects an instance in an edit history. The method comprises the steps of storing indications of the versions of the workflow, and displaying the versions of the workflow in a sequential manner to simulate animation of the edit history.

[0014] In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium contains instructions for controlling a data processing system to perform a method. The method comprises the steps of retrieving edits to a plan, and determining whether to display in a forward mode. When it is determined to display in the forward mode, the method further comprises the steps of removing the edits from the plan, displaying the plan, and for each of the edits, applying the edit to the plan, and displaying the plan. When it is determined not to display in the forward mode, the method further comprises the steps of displaying the plan, and for each of the edits, removing the edit from the plan, and displaying the plan.

[0015] In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium contains instructions for controlling a data processing system to perform a method. The method comprises the steps of retrieving edits to a workflow, and determining whether to display in a forward mode. When it is determined to display in the forward mode, the method further comprises the steps of removing the edits from the workflow, displaying the workflow, and for each of the edits, applying the edit to the workflow, and displaying the workflow. When it is determined not to display in the forward mode, the method further comprises the steps of displaying the workflow, and for each of the edits, removing the edit from the workflow, and displaying the workflow.

[0016] In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium contains instructions for controlling a data processing system to perform a method. The method comprises the steps of retrieving a plurality of plans generated from a workflow, and displaying each of the plans in a sequential manner to simulate the generation of the plans from the workflow.

[0017] In accordance with articles of manufacture consistent with the present invention, a computer-readable medium is provided. The computer-readable medium con-

tains instructions for controlling a data processing system to perform a method. The data processing system comprises a plan and the plan comprises a plurality of tasks. The method comprises the steps of displaying a graphical representation of the plan, wherein the graphical representation has portions that correspond to the tasks, retrieving edits to the plan, wherein each of the edits modifies a state of one of the plurality of tasks, and for each of the edits, applying the edit to the corresponding task of the plan, and displaying the portion of the graphical representation that corresponds to the edited task in a visually distinctive manner.

[0018] Other systems, methods, features and advantages of the present invention will be or will become apparent to one with skill in the art upon examination of the following figures and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate an implementation of the present invention and, together with the description, serve to explain the advantages and principles of the invention. In the drawings:

[0020] **FIG. 1** depicts a data processing system suitable for practicing methods and systems consistent with the present invention;

[0021] **FIG. 2** depicts an architectural overview of the workflow modeling and project planning integration tool used to perform methods and systems consistent with the present invention;

[0022] **FIG. 3** depicts a flow diagram illustrating the high-level process performed by the tool of **FIG. 2** in accordance with methods and systems consistent with the present invention;

[0023] **FIG. 4** depicts an exemplary document workflow modeled by an enterprise affiliate using the tool of **FIG. 2**;

[0024] **FIG. 5** depicts an exemplary task workflow modeled by an enterprise affiliate using the tool of **FIG. 2**;

[0025] **FIG. 6** depicts another exemplary workflow modeled by an enterprise affiliate using the tool of **FIG. 2**;

[0026] **FIG. 7** depicts a timeline of the task created from the workflow of **FIG. 4**;

[0027] **FIG. 8** depicts a timeline of the task created from the workflow of **FIG. 5**;

[0028] **FIG. 9** depicts a timeline of the task created from the workflow of **FIG. 6**;

[0029] **FIGS. 10-12** depict the execution of the plan depicted in **FIG. 7**;

[0030] **FIGS. 13-16** depict the execution of the plan depicted in **FIG. 8**;

[0031] **FIGS. 17-21** depict the execution of the plan depicted in **FIG. 9** following the default path;

[0032] **FIGS. 22-27** depict the execution of the plan depicted in **FIG. 9** following the non-default path;

- [0033] FIGS. 28A-C depict a flow diagram illustrating the creation or retrieval of a workflow by the tool of FIG. 2;
- [0034] FIG. 29 depicts an exemplary user interface of the tool of FIG. 2 used to begin creating or retrieving a workflow;
- [0035] FIG. 30 depicts an exemplary user interface of the tool of FIG. 2 used to enter the name of a new workflow group;
- [0036] FIG. 31 depicts an exemplary user interface of the tool of FIG. 2 used to begin creating a new workflow;
- [0037] FIG. 32 depicts an exemplary user interface of the tool of FIG. 2 used to enter the name of a new workflow;
- [0038] FIGS. 33A-C depict an exemplary workflow definition file produced by the tool of FIG. 2 for the workflow depicted in FIG. 6; FIG. 34 depicts an exemplary user interface of the tool of FIG. 2 used to manage a workflow;
- [0039] FIG. 35 depicts an exemplary user interface of the tool of FIG. 2 used to add a new role to a workflow;
- [0040] FIG. 36 depicts an exemplary user interface of the tool of FIG. 2 used to select an artifact type;
- [0041] FIG. 37 depicts an exemplary user interface of the tool of FIG. 2 used to enter condition properties for a document-oriented artifact;
- [0042] FIG. 38 depicts an exemplary user interface of the tool of FIG. 2 used to enter condition properties for a script-oriented artifact;
- [0043] FIG. 39 depicts an exemplary user interface of a script editor for the tool of FIG. 2;
- [0044] FIG. 40 depicts an exemplary user interface of the tool of FIG. 2 used to modify the properties of a workflow activity;
- [0045] FIGS. 41A and B depict a flow diagram illustrating the creation of a plan from a workflow;
- [0046] FIG. 42 depicts an exemplary user interface of the tool of FIG. 2 used to create a new plan group;
- [0047] FIG. 43 depicts an exemplary user interface of the tool of FIG. 2 displaying the available plan groups;
- [0048] FIG. 44 depicts an exemplary user interface of the tool of FIG. 2 used to enter a plan name;
- [0049] FIG. 45 depicts an exemplary user interface of the tool of FIG. 2 used to enter the working schedule;
- [0050] FIG. 46 depicts an exemplary user interface of the tool of FIG. 2 used to enter the scheduled start and end times for the plan;
- [0051] FIG. 47 depicts an exemplary workflow definition file produced by the tool of FIG. 2 for the workflow of FIG. 5 is created;
- [0052] FIG. 48 depicts an exemplary plan definition file created from the workflow definition file of FIG. 47;
- [0053] FIG. 49 depicts an exemplary user interface of the tool of FIG. 2 used to assign users to a plan;
- [0054] FIG. 50 depicts an exemplary user interface of the tool of FIG. 2 used to edit the properties of a plan;
- [0055] FIG. 51 depicts a timeline of the task created from the workflow of FIG. 5;
- [0056] FIG. 52 depicts an exemplary timeline of the tool of FIG. 2 used to activate a plan;
- [0057] FIG. 53 depicts a flow diagram illustrating the addition of a resource by the tool of FIG. 2;
- [0058] FIG. 54 depicts an exemplary user interface of the tool of FIG. 2 used to add a resource;
- [0059] FIG. 55 depicts an exemplary user interface of the tool of FIG. 2 used to receive LDAP access information;
- [0060] FIG. 56 depicts an exemplary resource file created by the tool of FIG. 2;
- [0061] FIG. 57 depicts a flow diagram illustrating the management of an activated plan;
- [0062] FIG. 58 depicts a timeline of the task created from the workflow of FIG. 5;
- [0063] FIG. 59 depicts an exemplary plan definition file created from the workflow of FIG. 5;
- [0064] FIGS. 60, 62, 64 and 66 depict the actual timeline showing the execution of the plan depicted in FIG. 58;
- [0065] FIGS. 61, 63, and 65 depict the properties of the executing tasks of FIGS. 62, 64, and 66;
- [0066] FIG. 67 depicts a flow diagram illustrating the modifications to the plan definition and task definition during the activation of a plan;
- [0067] FIGS. 68, 70, 72, 75, and 77 depict the activation of the first task of an exemplary plan created using the tool of FIG. 2;
- [0068] FIGS. 69, 71, 73, 74, 76, and 78 depict the task definition file for the first task of FIGS. 68, 70, and 72;
- [0069] FIG. 79 depicts a flow diagram illustrating the steps performed by the tool depicted in FIG. 2 to edit a plan;
- [0070] FIGS. 80A and B depict a flow diagram illustrating the steps performed by the tool depicted in FIG. 2 to sequentially displaying the versions of the plan definition file;
- [0071] FIG. 81 depicts a flow diagram illustrating the steps performed by the tool depicted in FIG. 2 to edit a workflow;
- [0072] FIG. 82 depicts an exemplary workflow created using the tool of FIG. 2;
- [0073] FIGS. 83-87 depict activity definition files corresponding to the activities depicted in FIG. 82;
- [0074] FIG. 88 depicts the workflow of FIG. 82 with a revision;
- [0075] FIGS. 89 and 90 depict activity definition files corresponding to the modified activities for the workflow depicted in FIG. 88;
- [0076] FIG. 91 depicts a flow diagram illustrating the steps performed by the tool depicted in FIG. 2 to create different plans from one workflow; and

[0077] FIGS. 92A and B depict a flow diagram illustrating the steps performed by the tool depicted in FIG. 2 to sequentially display the versions of the plan definition file.

DETAILED DESCRIPTION OF THE INVENTION

[0078] Methods and systems consistent with the present invention provide an integrated workflow modeling and project planning integration tool that improves the efficiency and reduces the operating cost of an enterprise or business conglomerate. Contrary to conventional tools that do not allow a user to integrate a workflow and a project plan, the integration tool allows a user to model a business process or workflow, to create and activate a project plan based on the workflow, and to track the progress of the activated project plan. The tool also allows the workflow to be reused to create more than one project plan based on the workflow. The tool simultaneously manages the execution of the plans. Moreover, the integration tool may include a virtual file system server, such as a Web-based "Distributed Authoring and Versioning" (WebDAV) server that operates as a virtual file system for computers on a network. With the WebDAV server, more than one user on different computer systems may view the same workflow or project plan, monitor the progress of an activated project plan, or simultaneously create and activate different plans from the same workflow.

System Overview

[0079] While methods and systems consistent with the present invention may apply to any enterprise in any industry, they will be further described below with reference to the software industry to provide clarity, consistency, and to demonstrate the invention as applied to one of the more difficult process industries. More particularly, methods and systems consistent with the present invention will be described with reference to a software development business process that is applicable to the software industry.

[0080] FIG. 1 depicts a data processing system 100 suitable for practicing methods and systems consistent with the present invention. Data processing system 100 includes a group of computers 102a, 104, and 106 that are connected via a network 108. Network 108 may be any known physical or wireless network capable of supporting a data transmission between two computer systems, such as a Local Area Network (LAN), a Wide Area Network (WAN), the Internet, or leased phone lines.

[0081] As further explained herein, computer 102a may actually be one of multiple computers (i.e., computers 102a and 102n) used by affiliates of an enterprise or business conglomerate to communicate with one another via network 108. The enterprise affiliates may be employees, managers, administrators, suppliers, customers, other computer applications, other computer systems, or other users within the enterprise who may need to create, view, or receive information regarding an activated project plan in accordance with methods and systems consistent with the present invention.

[0082] Each computer 102a, 104, and 106 includes a memory (110, 112, and 114, respectively), a secondary storage device (116, 118, and 120, respectively), an I/O device (122, 124, and 126, respectively), and a processor (128, 130, and 132, respectively). Memory 110 in computer

102a includes a Client Interface 134 to a Web-based "Distributed Authoring and Versioning" (WebDAV) server 140 in memory 112. Client Interface 134 has Process and Plan modules 136 that collectively allow an enterprise affiliate to create a reusable workflow and to create and activate a project plan based on the reusable workflow.

[0083] Memory 110 in computer 102a also includes a target processor interpreter, such as a Java™ Virtual Machine 138. To permit the Client Interface 134 to run on most any computer, the Client Interface 134 may be developed using the Java™ Programming Language. Thus, Client Interface 134 may include Java™ bytecodes. The Java™ Virtual Machine 138 interprets the Java™ bytecodes of the Client Interface 134 so that the Client Interface 134 may execute on computer 102a.

[0084] The WebDAV server 140 in memory 112 of computer 104 operates as a virtual file system for computers 102a, 102n, and 106 on the network 108. To operate as a virtual file system, WebDAV Server 140 communicates on the network 108 using the WebDAV protocol, and stores files on WebDAV storage 142. In one implementation, WebDAV storage 142 may be a known database, such as Oracle, DB2, MS Structured Query Language (SQL) storage, or any Java Database Connectivity (JDBC)-compliant database. In this implementation, WebDAV Server 140 includes a database management system (DBMS) or a JDBC interface to control access to the WebDAV storage 142.

[0085] In accordance with methods and systems consistent with the present invention, two separate enterprise affiliates using their respective Client Interfaces 134 on their respective computers 102a and 102n may independently request and view the same workflow or plan stored on WebDAV Storage 142. In addition, the Client Interface 134 allows any enterprise affiliate to create, delete, move, and copy workflows, project plans, and associated roles/resource lists on WebDAV server 140. Furthermore, properties of a process, a plan, or a task of a plan may be added, located, or changed on WebDAV Storage 142 by Client Interface 134 using known methods of the WebDAV protocol.

[0086] The WebDAV protocol is a set of known extensions to the standard HyperText Transfer protocol (HTTP) that allows enterprise affiliates using client computers 102a and 102n to collaboratively store, edit, and manage files remotely on a Web Server, such as WebDAV Server 140 using network 108. As known to one skilled in the art, HTTP defines how messages sent to or from a Web server on the Internet are formatted and transmitted. HTTP also defines what actions a Web server or Web browser of a computer on the Internet should take in response to various commands submitted as part of an HTTP message.

[0087] The WebDAV protocol defines a WebDAV resource to be a collection (e.g., a directory or folder on WebDAV Storage 142) or a collection member (e.g., a file or Web page on WebDAV Storage 142). Each WebDAV resource has a content file and properties associated with the content file. The properties include the creation date, the author, and the access rights for the WebDAV resource. The WebDAV protocol specifies the methods to create, delete, move, and copy a WebDAV resource. It also specifies the methods to add, find, or change a property of a WebDAV resource. The WebDAV protocol and the HTTP extensions that comprise the WebDAV protocol are more clearly

described in the following reference, which is incorporated herein by reference: HTTP Extensions For Distributed Authoring—WebDAV, RFC 2518, Standards Track, Proposed Standard, February 1999, available at <http://andrew2.andrew.cmu.edu/rfc/rfc2518.html>.

[0088] Memory 114 in computer 106 includes a Tool Server 144. The Tool Server 144 includes a WebDAV proxy 146 and Management Modules 148. WebDAV proxy 146 mediates communication between the Client Interface 134 and the WebDAV server 140. The messages or requests directed to the WebDAV server 140 from the Client Interface 134 are initially sent to the WebDAV proxy 146, as discussed in detail below. The WebDAV proxy 146 passes these messages and requests to the Management Modules 148. Each of the Management Modules 148 is configured to inform the WebDAV proxy 146 when a message or request has been serviced. If none of the Management Modules 148 services the message or request, then the WebDAV proxy 146 sends the message or request to the WebDAV Server 140 via the network 108. If, on the other hand, the Management Modules 148 are able to service the message or request, the message or request is not sent to the WebDAV Server 140. The types of messages or requests serviced by the Management Modules 148 include activating a project plan, notifying various enterprise affiliates assigned to each task of the plan, and managing the execution of the plan to the enterprise affiliates.

[0089] In another implementation, memory 114 in computer 106 includes a WebDAV servlet (not shown), which is an application designed to perform as a WebDAV Engine in lieu of WebDAV Server 140. The WebDAV servlet may be started by and executed within the Tool Server 144. In this implementation, WebDAV servlet is persistent. Thus, once WebDAV servlet is started, it stays in memory and can fulfill multiple requests. WebDAV servlet is configured to control access to WebDAV Storage 142, which may be included in secondary storage 120 in computer 106.

[0090] Memory 114 in computer 106 also includes a target processor interpreter, such as a Java™ Virtual Machine 150. As with the Client Interface 134 on computer 102a, the Tool Server 144 includes Java™ bytecodes that the Java Virtual Machine 150 interprets so that the Tool Server 144 may execute on computer 106.

[0091] In another implementation, the data processing system 100 may operate in a local host configuration rather than across the network 108. In this implementation, the memory 110 of computer 102a may include the Tool Server 144 and the WebDAV Server 140. In addition, the secondary storage device 116 may include the WebDAV Storage 142.

[0092] Although aspects of the present invention are described as being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on or read from other types of computer-readable media, such as secondary storage devices, like hard disks, floppy disks, or CD-ROM; a carrier wave from a network, such as Internet; or other forms of RAM or ROM.

[0093] FIG. 2 depicts a functional architectural overview of the workflow modeling and project planning integration tool 200 used to integrate workflow modeling and project planning. As shown in FIG. 2, the tool 200 includes the Client Interface 134 as well as the Tool Server 144. Although

part of the same tool 200, the Client Interface 134 and the Tool Server 144 may be located on different computer systems, as discussed above.

[0094] The Client Interface 134 includes a Virtual File System (“VFS”) Interface 202 that is configured to allow the Client Interface 134 to connect to the secondary storage device 116 for local file access or to connect to the WebDAV Storage 142 via the WebDAV proxy 146 for virtual file access. To allow the WebDAV proxy 146 to mediate communication between the Client Interface 134 and the WebDAV Storage 142, the VFS Interface 202 is configured to send the virtual file access requests from the Client Interface 134 to a Uniform Resource Locator (URL) or network address for the WebDAV proxy 146. For example, the URL for the WebDAV proxy 146 may be “<http://www.ToolServer.com/WebDAVproxy>.” A URL typically consists of an access protocol (e.g., http), a domain name (e.g., www.ToolServer.com), and, optionally, the path to a file or resource residing on that server (e.g., WebDAVproxy). If the Tool Server 144, where the WebDAV proxy 146 is located, has an IP address of 192.168.5.1 and an assigned port address of 8088, then the URL for the WebDAV proxy translates to “<http://192.168.5.1:8088/WebDAVproxy>.” As discussed above, the VFS Interface 202 initially sends the requests that the Client Interface 134 directs to the WebDAV Storage 142 to the WebDAV proxy 146. The WebDAV proxy 146 sends these requests to the Management Modules 148. After the Management Modules 148 review these requests, the WebDAV proxy 146 sends the request to the WebDAV server 140 if the Management Modules 148 do not respond to the requests from the Client Interface 134. If the request is to be sent to the WebDAV server 140, the Tool Server 144 directs the request to a URL or network address for the WebDAV server 140.

[0095] The Client Interface 134 also includes a module loader 204 to load the Process and Plan modules 136. As one skilled in the art will appreciate, Client Interface 134 may be developed so that the functionality provided by Process and Plan modules 136 is not loaded by a known module loader 204, but integrally incorporated within the element corresponding to the Client Interface 134. The Process and Plan modules 136 produce the requests to store or modify the various client files on the WebDAV storage 142. As further described below, the various types of client files include a condition model, a user profile, a resource profile, a workflow definition file, and a plan definition file. Each of these files has properties defined in accordance with the WebDAV protocol. The various types of client files follow a schema or document type definition that is known to the Tool Server 144 so that the Tool Server 144 can identify the type of client file sent by the Client Interface 134 and intercepted by the WebDAV Proxy 146. In addition, each type of client file has a unique identifier, such as a URL network address, which the Tool Server 144 may use to locate the associated client file for processing. The various types of client files are discussed in context with the general description of the Process and Plan modules 136 and also further discussed with the implementation details of creating a workflow and a project plan from the workflow. Although XML files are used to represent the client files used with methods and systems consistent with the present invention, one skilled in the art will recognize that any file type can be used to represent the client files.

[0096] The Process and Plan Modules 136 include a Resource Manager Module 206, an Activity I/O Condition Designer Module 208, a Process Designer Module 210, a Project Plan Manager Module 212, and a Task Tracker Module 214. The Resource Manager Module 206 allows an enterprise affiliate with system administrative privileges or permissions, such as a project manager, to create, modify, and store a user profile for an enterprise affiliate. The user profile identifies the access control rights that are associated with the enterprise affiliate, such as whether the enterprise affiliate may create or edit or delete a project plan based on a workflow or whether the enterprise affiliate is limited to viewing an existing workflow or plan. When the Client Interface 134 sends a request to the WebDAV Server 140 to store the user profile, the Client Interface 134 may specify that the user profile be stored with a unique identifier so that the Tool Server 144 may later locate the user profile for further processing. For example, the Client Interface 134 may request that the unique identifier be a location or URL where the user profile is to be stored on the WebDAV Storage 142. If the unique identifier is stored as a property of the user profile on the WebDAV storage 142, the Client Interface 134 sends a request to the WebDAV Server 140 to set the value of the property.

[0097] The Resource Manager Module 206 also allows an enterprise affiliate to create, modify, and store the role profiles that may be assigned to an activity of a workflow that is modeled using the tool 200. The role profile identifies a group of resources that may be assigned to complete a task created from the activity. The role profile is a type of client file that the Client Interface 134 may store on WebDAV storage 142 with a unique identifier (e.g., a URL for the role profile) to locate the role profile at a later time. A role profile may include a Rolename that represents a "capability" or "skill set" for the role. For example, using methods and systems consistent with the present invention, an enterprise affiliate may identify one of the following Rolenames to the Resource Manager Module 206 so that the associated role profiles are later available to assign when defining a software development process: Manager, Analyst, Software Architect, Software Developer, Tester, Hardware Architect, and Editor.

[0098] In addition to the above, the Resource Manager Module 206 further allows an enterprise affiliate to create, modify, and store the resource profiles (e.g., the person, equipment, or systems, such as a development facility) that may be assigned to a task of a plan created from a workflow. The resource profile includes a resource ID and a unique identifier for the role profile so that the Client Interface 134 may communicate to the Tool Server 144 that the identified resource has skills or capabilities corresponding to the role profile. For example, when the resource is a person, the Tool Server 144 may recognize that the person can play a given role (e.g., Analyst) in a specific activity (e.g., Requirements Analysis) in a workflow (e.g., Software Development Process) based on the skills or capabilities required by the role assigned to the activity to be performed.

[0099] The Activity I/O Condition Designer Module 208 allows an enterprise affiliate, such as a manager, to define a condition model, i.e., an input condition or an output condition, for an activity of a workflow. The Activity I/O Condition Designer Module 208 stores the condition model with a unique identifier so that the Tool Server 144 may later locate the condition model for processing, such as when a task of a plan is created from the activity of the workflow, as explained below.

[0100] As discussed above, there are two types of workflows: a document workflow and a task workflow. Similarly, there are two types of conditions: a document-type condition and a logic-type condition. The Activity I/O Condition Designer Module 208 allows the enterprise affiliate to create a condition model based on one of these two condition types. The Activity I/O Condition Designer Module 208 also allows the enterprise affiliate to assign a document-type condition model or a logic-type condition model to an activity when creating the activity in a workflow. Each document-type and logic-type condition model has properties defined by the enterprise affiliate that created the respective condition model using the Activity I/O Condition Designer Module 208.

[0101] The properties of the document-type condition model include a location property (e.g., a URL) identifying the location of the document or artifact being monitored. Thus, when executing a task based on an activity, the Client Interface 134 uses the location property to notify the resource responsible for the task where to find the document or artifact so that the resource may complete its task.

[0102] Another property of the document-type condition model is a state property that indicates the allowable states of the document or artifact. For example, the document may have the states "DRAFT" and "APPROVED." When creating the workflow, the enterprise affiliate assigns one of these allowable states as a condition for entry into or exit from the activity (or the task created from the activity). When the task is activated, the Workflow Engine 222 evaluates whether the state property of the document condition satisfies the input or output condition of the activated task before starting or closing the task.

[0103] When creating a logic-type condition model, Activity I/O Condition Designer Module 208 allows the enterprise affiliate to define the properties shown in Table 1.

TABLE 1

Property	Description
Name	The name used to identify the condition.
Description	A description of the condition.
When to Check	This section identifies when and/or how often the condition should be checked.
Abs. Time	The condition is checked when this absolute time (calendar time) arrives.
Period	Integer expression in Javascript that defines the periodicity of condition check, where a "1" means once a minute. (If absolute time is also specified, then the condition should be checked when the absolute time arrives and periodically thereafter.)
URL Change	The condition is checked after URL undergoes a property or content change.
Task Change	The condition is checked when the task that is specified during plan creation changes its state (e.g., starts, finishes).
Any Request	The condition is checked when any HTTP request is detected.
Script	The script to run when the condition is met. The script must return "true" or "false" (a Boolean). Script is an extensible method for users to enter in ad hoc logic.

[0104] When a plan is created from a workflow, the Client Interface 134 uses the logic-type condition model to generate a logic-type condition for entry/exit and the script (e.g.,

logic element to be performed to determine if the condition is met. For example, the enterprise affiliate may indicate to the Activity I/O Condition Designer Module 208 that the condition is to check if the task is complete and that the logic to be performed is to check the status property of the task. In this case, the user resource assigned to this task must notify the Client Interface 134 that the task is complete. In another example the enterprise affiliate may indicate to the Activity I/O Condition Designer Module 208 that the condition is to check if the task is complete and that the logic to be performed is to check for the existence of a file in a specific directory folder on WebDAV Storage 142 in order to determine if the task is complete. In this case, the user or resource assigned to this task must create or move a file into the specific directory folder to indicate that the task is complete.

[0105] The Process Designer Module 210 allows an enterprise affiliate to create, modify, and store a workflow. When the enterprise affiliate indicates to Process Designer Module 210 that the modeled process is to be saved, Process Designer Module 210 produces a workflow definition file based on the modeled workflow object. Client Interface 134 then sends as the workflow definition file as a client file to WebDAV Server 140 to be stored on WebDAV Storage 142. Each workflow definition file produced by Process Designer Module 210 includes a unique identifier (e.g., a URL for the workflow definition file) so that Tool Server 144 may later locate the workflow definition file corresponding to the modeled workflow for further processing.

[0106] Project Plan Manager Module 212 allows an enterprise affiliate to create and activate a project plan from a workflow definition file. In general, upon request to create a project plan, Project Plan Manager Module 212 sends a query message to the WebDAV Server 140 for the workflow definition files contained in WebDAV Storage 142. As further described below, Project Plan Manager Module 212 receives the workflow definition files, allows the enterprise affiliate to select one of the workflow definition files to create a project plan, and then produces a plan definition file based on the selected workflow definition file. When instructed to save the plan by the enterprise affiliate, Project Plan Manager Module 212 sends the plan definition file as a client file to WebDAV Server 140 to be stored on WebDAV Storage 142. Each plan definition file produced by Process Designer Module 210 includes a unique identifier (e.g., a URL for the plan definition file) so that Tool Server 144 may later locate the workflow definition file corresponding to the modeled workflow for further processing.

[0107] The Task Tracker Module 214 allows an enterprise affiliate to view the tasks of an activated project plan that are assigned to a specific resource, to activate or start a task of the project plan (e.g., indicate actual start time to Client Interface 134), to open or check-out a document artifact needed to accomplish the task, to close or check-in the document artifact after accomplishing the task, and to indicate that the task is completed.

[0108] The Tool Server 144 includes a module loader 216 to load the Management Modules 148. Similar to the Client Interface 134, the Tool Server 144 may be developed so that the functionality provided by Management Modules 148 is not loaded by a known module loader 216, but integrally incorporated within the element corresponding to the Tool

Server 144. Management Modules 148 include a User Authorization Module 218, a Resource/Role Management Module 220, and a Workflow Engine 222. The Workflow Engine 222 includes a Project Plan Management Module 224 and a Project Task Management Module 226.

[0109] When the Client Interface 134 requests access to a client file on the WebDAV Storage 142, the User Authorization Module 218 verifies that the enterprise affiliate making the request has a user profile on the WebDAV Storage 142 with the proper authorization or permission to access the requested client file. The User Authorization Module 218 may be connected to a Light Directory Access Protocol (LDAP) Import Module 228, which follows a known LDAP protocol to allow the User Authorization Module 218 to obtain existing user profiles from another computer on network 108. As known to those skilled in the art, an LDAP protocol is based on "entries," where an entry is a collection of attributes that have a "distinguished name" (DN). According to the LDAP protocol, directory entries are arranged in a hierarchical tree-like structure that reflects political, geographic, and/or organizational boundaries. For example, entries representing countries may appear at the top of the tree. The entries below the countries may represent states or national organizations. Below the states or national organizations may be entries representing people (e.g., user profiles), organizational units, printers, documents, or any other accessible entity. Each level in the hierarchical tree-like structure for the directory entries may be identified by a known standardized keyword, such as "CN" for the common name of the entry (e.g., user profile), "L" for locality name, "ST" for state or province name, "O" for organization name, "OU" for organizational unit name, and "C" for country name. The LDAP Import Module 228 uses a DN to refer to the entry unambiguously via a concatenation of the hierarchical tree-like structure. After user profiles are retrieved by the User Authorization Module 218 via the LDAP import module 228, the user profiles may then be stored on the WebDAV Storage 142 by a request from the Client Interface 134.

[0110] The Resource/Role Management Module 220 reviews requests from an enterprise affiliate to assign a resource to a plan (e.g., to assign a user to a task of the plan). The Resource/Role Management Module 220 may check the resource profile corresponding to the assigned resource on the WebDAV Storage 142 to verify that the resource is not overloaded. For example, the Resource/Role Management Module 220 determines whether a resource is already assigned to another task in another plan during the same time frame, thus preventing it from being able to complete one of the tasks to which it is assigned. The Resource/Role Management Module 220 may also be connected to the LDAP Import Module 228 to allow the Resource/Role Management Module 220 to obtain existing resource profiles from another computer on network 108. The resource profiles may also be stored on WebDAV Storage 142 by a request from Client Interface 134.

[0111] The Workflow Engine 222 reviews requests to activate, deactivate, or update a plan. For example, a request to update a plan occurs if the enterprise affiliate who is an owner of a task indicates in its request that the task is complete. The Workflow Engine 222 also manages the execution of the activated plans.

High-Level Process

[0112] FIG. 3 depicts a flow diagram illustrating the high-level process performed by the workflow modeling and project planning integration tool in accordance with methods and systems consistent with the present invention. Initially, the tool creates or retrieves a workflow (step 302). The tool then displays the workflow (step 304). The workflow comprises a set of activities that represents the steps to be performed as part of a plan executed from the workflow. Each activity has an activity description and at least one role responsible for the activity. The activity description indicates what step is to be performed by the role.

[0113] There are two types of workflows: a document workflow and a task workflow. In a document workflow, the state of one document (or, more generally, any item or artifact) is monitored by the activities of the workflow. Thus, a document workflow cannot usually have parallel activities, which would require the parallel activities to monitor the states of more than one artifact or would require the parallel activities to monitor different states of the same artifact simultaneously. The document is in one state at a time. FIG. 4 depicts an exemplary document workflow 400. As shown, the workflow 400 includes a start element 402, an end element 404, and two activities, "Step 1"406 and "Step 2"408. Because "Step 1"406 occurs directly before "Step 2"408, "Step 1"406 is the "predecessor activity" to "Step 2"408. Similarly, "Step 2"408 is the "successor activity" to "Step 1"406. The workflow 400 is used to monitor the state of Artifact 410. In particular, in "Step 1"406, the state of Artifact 410 is "State 1"412, in "Step 2"408, the state of Artifact 410 is "State 2"414, and at the end 404 of the workflow, the state of Artifact 410 is "Complete"416.

[0114] A task workflow, on the other hand, typically has no limitations regarding the number of artifacts that may be monitored or modified by each activity of the workflow to achieve or contribute to the business process goal, such as an auditing process that determines if multiple accounts are balanced properly. FIG. 5 depicts an exemplary task workflow 500. The task workflow 500 includes a start element 502, an end element 504, two serial activities 506 and 508 and two parallel activities 510 and 512. The workflow also includes two synchronizers 514 and 516, which are used to connect the ends of parallel activities. Contrary to the document workflow, the task workflow allows for parallel activities.

[0115] Another exemplary workflow 600 is depicted in FIG. 6. The workflow 600 includes a start element 602 and an end element 604. The first activity of the workflow 600 is "Get Parts"606, which is followed by a logic activity, "L or Rt Handed?"608. Logic activities have two successor activities: a "default activity" and a "non-default activity." As the name implies, the workflow generally follows the path of the default activity unless a condition is met in the logic activity, as discussed in detail below. In FIG. 6, the default activity is "Right"610. The non-default activity is "Left"612, which is followed by another activity "Left Special"614. The default path is represented as a solid connector 616 while the non-default path is represented as a dotted connector 618. One skilled in the art, however, will recognize that any visible difference in the connectors, e.g., a change in type, color, shading, labeling, etc., may be used to represent both the default path as well as the non-default

path. Both the default activity 610 and the non-default activities 612 and 614 are followed by another activity, "Complete Assembly"620. In addition, though we show only two paths (616 & 618) out of the decision block 608, there could be any number of exit paths (not shown).

[0116] Returning to FIG. 3, the next step performed by the tool is to create a plan from the workflow (step 306). Each activity in the default path of the workflow generally corresponds to a task in the plan. The task identifies the scheduled start and stop times for the task. The tool then displays the plan (step 308). For example, the plan created from the workflow 400 depicted in FIG. 4 is shown in FIG. 7. The plan 700 includes two tasks 702 and 704 that correspond to the two activities 406 and 408 from the workflow 400. The first task 702 is scheduled to begin at 9 a.m. 706 on Aug. 1, 2001 (not shown), and end at 6 p.m. 708 on the same day. The second task 704 is scheduled to begin at 9 a.m. 710 on Aug. 2, 2001 (712) and end at 5 p.m. 714 on the same day.

[0117] The plan 800 created from the workflow 500 depicted in FIG. 5 is shown in FIG. 8. The plan 800 includes two serial tasks 802 and 804 that correspond to the two serial activities 506 and 508 from the workflow 500. The plan 800 also includes two parallel tasks 806 and 808 that correspond to the two parallel activities 510 and 512 from the workflow 500. As shown in FIG. 8, "Serial 1" task 802 is scheduled to begin at 9 a.m. 810 on Aug. 1, 2001 (812) and end at 5:30 p.m. 814 on the same day. The parallel tasks 806 and 808 are scheduled to start at the completion of the "Serial 1" task 802, and are scheduled to end at 6 p.m. 816 on Aug. 2, 2001 (818). The "Serial 2" task 804 is scheduled to begin upon completion of the parallel tasks 806 and 808 and is scheduled to end at 6 p.m. 820 on Aug. 3, 2001 (822).

[0118] The plan 900 created from the workflow 600 depicted in FIG. 6 is shown in FIG. 9. The plan 900 includes a task 902 corresponding to the activity "Get Parts"606, followed by a task 904 corresponding to the activity "L or Rt Handed?"608. The following task 906 corresponds to the activity "Right"610. The final task 908 corresponds to the activity "Complete Assembly"620. The plan 900 depicts the default path, and does not include any of the tasks corresponding to the non-default path. Although the start and end times are not depicted in the plan 900 shown in FIG. 9, each task has a scheduled start and stop time. In addition, the tool 200 requires that an enterprise affiliate assign a resource to each task, as described below.

[0119] Returning to the high-level process of FIG. 3, the tool then activates the plan (step 310). Next, the tool manages the execution of the activated plan (step 312). The tool also modifies the display of the plan as each task is executed (step 314). The tool then determines whether the execution of the plan is complete (step 316). If the execution of the plan is complete, processing ends. Otherwise, processing continues to step 312.

[0120] For the exemplary plan 700 depicted in FIG. 7, upon activation, the first task 702 begins execution. The tool depicts the executing task 1002 by darkening the outer borders of the block representing the task 1002, as depicted in the plan 1000 shown in FIG. 10. After completion of the task, the tool depicts the executed task 1102 as a darkened block in the plan 1100, as shown in FIG. 11. At this point, the second task 1104 begins execution, as indicated by the

darkened outer borders of the task 1104. Finally, after both tasks 1102 and 1104 of the plan 1100 have been executed, both tasks 1202 and 1204 are depicted as darkened blocks in the plan 1200, as shown in FIG. 12. In this embodiment, the tool represents an executing task with darkened borders and represents an executed task as a darkened block. One skilled in the art, however, will recognize that any visible change in the blocks representing the tasks, e.g., a change in shape, color, shading, labeling, etc., may be used to represent the tasks in their various states. For example, in another implementation, color may be used to indicate active tasks; for example a gray rectangle may be used behind the task to indicate an actual activity since the actual dates may not coincide with the dates of the planned task. Thus, the representation of the tasks used in the methods, systems, and articles of manufacture consistent with the present invention are not limited to those used in the present embodiment.

[0121] The activation and execution of the tasks of the plan 800 depicted in FIG. 8 are shown in FIGS. 13-16. FIG. 13 depicts the state of the plan 1300 while the "Serial 1" task 1302 is executing. FIG. 14 depicts the state of the plan 1400 after execution of the "Serial 1" task 1402, while the "Parallel 1" and the "Parallel 2" tasks 1404 and 1406 are executing. FIG. 15 depicts the state of the plan 1500 after execution of the "Serial 1" task 1502 and the "Parallel 1" and the "Parallel 2" tasks 1504 and 1506, while the "Serial 2" task 1508 is executing. Finally, FIG. 16 depicts the state of the plan 1600 after execution of the tasks 1602, 1604, 1606, and 1608.

[0122] As discussed above, FIG. 9 represents a plan 900 created from a workflow 600 having a logic block 608. The activation and execution of the tasks of the plan 900 following the default path are shown in FIGS. 17-21, while the activation and execution of the tasks of the plan 900 following the non-default path are shown in FIGS. 22-27.

[0123] FIG. 17 depicts the state of the plan 1700 while the "Get Parts" task 1702 is executing. FIG. 18 depicts the state of the plan 1800 after the execution of the "Get Parts" task 1802, while the "L Or Rt Handed?" logic task 1804 is executing. The logic task may pop up a dialog (not shown) to prompt the resource assigned to this task to provide an answer for this "left or right-handed" question. In addition, the tool allows the question to be "answered" by running a logic script. This script may examine properties of an indicated artifact or it may execute a separate program on a separate system to compute the answer. Upon selection of the default path, the plan 1900 shown in FIG. 19 depicts both the "Get Parts" task 1902 and the "L Or Rt Handed?" logic task 1904 in executed states, while the "Right" task 1906 is depicted in an executing state. After the execution of the "Right" task 1906 is complete, the state of the plan 2000 is depicted in FIG. 20 with the "Get Parts" task 2002, the "L Or Rt Handed?" logic task 2004, and the "Right" task 2006 in executed states and with the "Complete Assembly" task 2008 in an executing state. Finally, upon completion of the "Complete Assembly" task 2008, the state of the plan 2100 after execution of the tasks 2102, 2104, 2106, and 2108 is complete is depicted in FIG. 21.

[0124] Alternatively, if the non-default path is to be chosen, the execution of the plan is initially the same as when the default path is chosen. Thus, as depicted in FIG. 22, the plan 2200 begins with the execution of the "Get Parts" task

2202. After completion of the "Get Parts" task 2202, the plan 2300 shown in FIG. 23 depicts the "Get Parts" task 2302 in an executed state while the "L Or Rt Handed?" task 2304 is shown in an executing state. At this point, the resource assigned to choose the default or the non-default path chooses the non-default path, thus completing the execution of the "L Or Rt Handed?" task 2404, as indicated in FIG. 24. Upon selection of the non-default path, the tool 200 modifies the plan 2400 to correspond to the non-default path of the corresponding workflow. The plan 2400 depicts the tasks included in the non-default path. Thus, the plan 2400 includes the "Left" and "Left Special" tasks 2406 and 2408 rather than the "Right" task 2306, which is depicted in FIG. 23 before the non-default path was chosen. As shown in FIG. 24, the "Left" task 2406 is executing. FIG. 25 depicts the plan 2500 after the "Get Parts" task 2502, the "L Or Rt Handed?" logic task 2504, and the "Left" task 2506 have been executed, while the "Left Special" task 2508 is executing. Continuing with the execution of the plan, FIG. 26 depicts the state of the plan 2600 after the "Get Parts" task 2602, the "L Or Rt Handed?" logic task 2604, the "Left" task 2606, and the "Left Special" task 2608 are done executing, while the "Complete Assembly" task 2610 is executing. Finally, FIG. 27 depicts the state of the plan 2700 after completion of the tasks 2702, 2704, 2706, 2708, and 2710.

Retrieving Or Creating A Workflow

[0125] FIGS. 28A-C depict a flow diagram illustrating an exemplary process for retrieving or creating a workflow, i.e., step 302 in FIG. 3. Initially, the tool 200 determines whether to use an existing process or workflow group (step 2802). A workflow group is a collection of workflows (e.g., a directory or folder containing the collection of workflows) created by the Client Interface 134 on WebDAV Storage 142. Each workflow group is created by the Client Interface 134 on WebDAV Storage 142 with the "workflow group" property as explained further below. When creating a workflow, the Client Interface 134 allows the enterprise affiliate to store the workflow within an identified workflow group so that any enterprise affiliate using the Client Interface 134 is able to easily identify related workflows using a hierarchical relationship. For example, software-related workflows may be stored within the same workflow group so that an enterprise affiliate is able to quickly locate a desired workflow in order to create a corresponding plan using the Client Interface 134. One skilled in the art will appreciate that Client Interface 134 may store a workflow on WebDAV Storage 142 without associating the workflow with a workflow group.

[0126] The tool 200 receives user input from an enterprise affiliate with system administrative privileges or permissions, such as a process designer or a project manager, to determine whether to retrieve an existing workflow group or to create a new workflow group. If the tool 200 determines that it will use an existing workflow group, the tool 200 receives an identification of the workflow group from the enterprise affiliate (step 2804). In one implementation, the Client Interface 134 may retrieve the identifications for the workflow groups on the WebDAV Storage 142 by requesting that the folders or directories on WebDAV Storage 142 having a "workflow" property be returned by the WebDAV Server 140. The Client Interface may use any known method in accordance with WebDAV protocol to request that the WebDAV Server 140 return any directory or folder on

WebDAV Storage 142 that corresponds to a workflow group. The tool 200 may then display the available workflow groups to allow the enterprise affiliate to select one of the available workflow groups. For example, as shown on the user interface 2900 depicted in FIG. 29, the tool 200 may display a hierarchical view 2902 of an identified workflow group 2904 stored on the root directory 2906 of WebDAV Storage 142. Alternatively, the enterprise affiliate may enter the identification of the desired workflow group to the tool 200 for retrieval. Using the identification, the tool 200 then retrieves the workflow group (step 2806).

[0127] If the tool 200 determines that a new workflow group will be created, the tool 200 receives the name of the workflow group from the enterprise affiliate (step 2808). For example, the enterprise affiliate may request a new workflow group by clicking on “Process Designer” button 2908 of the user interface 2900 depicted in FIG. 29. The enterprise affiliate may, alternatively, use any known data input technique, such as an icon or keyboard input, to indicate the request to the tool 200. Upon selecting the “Process Designer” button 2908, the tool 200 displays an exemplary user interface 3000 depicted in FIG. 30 for receiving a new workflow group identification 3002 via keyboard input from an enterprise affiliate using computer 102a or 102n.

[0128] After receiving the new workflow group identification, the tool 200 creates a new workflow group in storage (step 2810). For example, the tool 200 may create the workflow group on WebDAV Storage 142. To generate a new workflow group on WebDAV Storage 142, the Client Interface 134 sends the WebDAV Server 140 a request to create a new collection or folder on WebDAV Storage 142 with the same identification as the new workflow group identification 3002. In accordance with WebDAV protocol, the Client Interface 134 receives a response from the WebDAV Server 140 confirming that the new workflow group folder was created on WebDAV Storage 142. As previously discussed, when a new collection or folder is created using the WebDAV protocol, the WebDAV properties (e.g., “date of creation,” “property name” and “lockdiscovery” properties) are created and stored in association with the new directory by the WebDAV Server 140. Thus, when generating the new workflow group, the Client Interface 134 also sets the “property name” of the new workflow group to be “workflow group” so that the Client Interface may subsequently use known WebDAV methods, such as “PropFind,” to retrieve the identification of each workflow group on WebDAV Storage 142.

[0129] After retrieving an existing workflow group or creating a new workflow group, the tool 200 determines whether to use an existing workflow (step 2812). The tool 200 receives user input from an enterprise affiliate with appropriate privileges or permissions to determine whether to retrieve an existing workflow or to create a new workflow. If the tool 200 determines that it will use an existing workflow, the tool 200 receives an identification of the workflow from the enterprise affiliate (step 2814). In one implementation, the Client Interface 134 may retrieve the identifications for the workflows in the selected workflow group and display the available workflows to allow the enterprise affiliate to select one of the available workflows. Alternatively, the enterprise affiliate may enter the identi-

fication of the desired workflow to the tool 200 for retrieval. Using the identification, the tool 200 then retrieves the workflow (step 2816).

[0130] If the tool 200 determines that a new workflow will be created, the tool 200 receives the name of the workflow from the enterprise affiliate (step 2818). For example, the enterprise affiliate may request a new workflow by clicking on the desired workflow group 3102 and selecting the “New Process” option 3104 from a pull-down menu 3106 on the user interface 3100 depicted in FIG. 31. The enterprise affiliate may, alternatively, use any known data input technique, such as an icon or keyboard input, to indicate the request to the tool 200. Upon selecting the “New Process” option 3104, the tool 200 may display the exemplary dialog box 3200 depicted in FIG. 32 to the enterprise affiliate. The enterprise affiliate may then enter the name of a new workflow 3202. After receiving the name for the workflow, the tool 200 creates the workflow in storage (step 2820).

[0131] FIGS. 33A-C depict an exemplary workflow definition file 3300 that is produced by the tool 200 when the workflow 600 depicted in FIG. 6 is created. The name 3302 of the workflow, “Logic Branch Project,” is identified in the workflow definition file 3300. Also, as shown in the definition file 3300, the workflow 600 does not have a workflow group 3304. The element 3306 in the workflow definition file 3300 represents the “Get Parts” activity 606. Similarly, the element 3308 (FIG. 33C) represents the “L or Rt Handed?” logic activity 608, the element 3310 represents the “Right” activity 610, the element 3312 represents the “Left” activity 612, the element 3314 represents the “Left Special” activity 614, and the element 3316 represents the “Complete Assembly” activity 620. The start element 602 is represented by the element 3318, and the end element 604 is represented by the element 3320.

[0132] The next step performed by the tool 200 is to receive an indication of the type of activity to be created for the workflow (step 2822 in FIG. 28B). As discussed above, the activity may be a standard activity or a logic activity. For example, the workflow 3402 depicted in the user interface 3400 of FIG. 34 includes five standard activities 3404, 3406, 3408, 3410, and 3412. The workflow 3402 also includes one logic activity 3414. The selection of the type of activity may be made by clicking on the icon for a standard activity 3416 or the icon for the logic activity 3418. Alternatively, any known data input technique, such as a pull-down menu or keyboard input, may be used to select the type of activity.

[0133] After receiving an indication of the type of activity, the tool 200 receives the name of the activity (step 2824). The names of the activities depicted in the workflow 3402 are included with the activity. Thus, the name of activity 3404 is “Assignment,” the name of activity 3406 is “Analysis,” etc.

[0134] Returning to the example workflow 600 depicted in FIG. 6, the name of the first activity 606 is “Get Parts,” which is identified by the element 3322 in the workflow definition file 3300 of FIG. 33. Similarly, the name of the logic activity 608 is “L or Rt Handed?,” which is identified by the element 3324. The name of the activity 610 is “Right,” as identified by the element 3326. The name of the activity 612 is “Left,” as identified by the element 3328. The name of the activity 614 is “Left Special,” as identified by the element 3330. Finally, the name of the activity 620 is “Complete Activity,” as identified by the element 3332.

[0135] After receiving a name for the activity, the tool 200 receives an indication of the role responsible for the activity (step 2826). As discussed above, the Client Interface (via Resource Manager Module 206) allows an enterprise affiliate to identify a role or role profile that may be assigned to an activity of the workflow. A role profile includes a Role-name that represents a “capability” or “skill set,” which is needed to perform a task of a plan created from the workflow, where the task corresponds to the activity of the workflow. For example, FIG. 35 depicts a user interface 3500 displayed by the Client Interface to receive a role profile. In the implementation shown in FIG. 35, the Client Interface receives a Rolename 3502 (e.g., “Project Manager”) for the role profile via the enterprise affiliate clicking on an “Add” button 3504 and then entering Rolename 3502 in a dialog box 3506 that is displayed by the Client Interface. In another implementation, the Client Interface may also receive as additional entries (not shown) to dialog box 3506 a skill and an associated skill level for Rolename 3502 as part of this role profile. For example, the enterprise affiliate may indicate to the Client Interface via the additional entries to dialog box 3506 that the Rolename 3502 of “Project Manager” be associated with a skill entitled “Object-oriented software programming” and with a skill strength of “7” on a scale of 10. Assuming an enterprise affiliate is developing a workflow for producing a software development tool, the enterprise affiliate may assign to activities in the workflow the “Project Manager” role profile with this skill and skill level. Thus, when a plan is created from this workflow, a resource having the appropriate skill and skill level will automatically be assigned by the Client Interface to tasks corresponding to the activities with the “Project Manager” role assignment.

[0136] The tool 200 stores the role profiles in association with the selected workflow activity on WebDAV Storage 142. The tool 200 saves significant costs in developing multiple workflows by allowing the enterprise affiliate to store the role profiles in association with the selected workflow activity on WebDAV Storage 142 so that the role profiles may be available for the enterprise affiliate to assign to an activity of another workflow that is also related to the selected workflow activity. In one implementation, the Client Interface stores the role profiles in a single role definition file (not shown) on WebDAV Storage 142. In another implementation, the Client Interface stores the role profiles in separate files (not shown) on WebDAV Storage 142. Each separate file has a name that is the same as the received Rolename 3502. In this implementation, using known WebDAV protocol, the Client Interface defines an associated WebDAV property having a common name, such as “role profile,” so that the Client Interface may later retrieve the role profiles stored on WebDAV storage.

[0137] The role profiles may also be stored with the workflow definition file. As shown in the workflow definition file 3300 depicted in FIG. 33, the role profile 3334 for the “Get Parts” activity 606 indicates that the role responsible for the activity is “Assembler”3336. Similarly, the role profile 3338 for the “L or Rt Handed?” activity 608 indicates that the role responsible for the activity is “Assembler”3340. The role profile 3342 for the “Right” activity 610 indicates that the role responsible for the activity is “Assembler”3344. The role profile 3346 for the “Left” activity 612 indicates that the role responsible for the activity is “Assembler”3348. The role profile 3350 for the “Left Special” activity 614

indicates that the role responsible for the activity is “Assembler”3352. Finally, the role profile 3354 for the “Complete Assembly” activity 620 indicates that the role responsible for the activity is “Assembler”3356.

[0138] The next step performed by the tool 200 is to determine whether the activity has any predecessor activities (step 2828). If the activity does have a predecessor activity, the tool 200 receives an indication of the predecessor activities from the workflow definition file (step 2830). After checking for any predecessor activities and/or receiving the predecessor activities, the tool 200 determines whether the activity has any successor activities (step 2832). If the activity has a successor activity, the tool 200 receives an indication of the successor activities from the workflow definition file (step 2834). In the user interface 3400 of FIG. 34, the “Path” icon 3420 is used to connect the predecessor activity to the successor activity. For example, in the workflow 3402, a path 3422 was drawn from the “Assignment” activity 3404 to the “Analysis” activity 3406. Thus, the “Assignment” activity 3404 is the predecessor activity to the “Analysis” activity 3406, and the “Analysis” activity 3406 is the successor activity to the “Assignment” activity 3404. Alternatively, a “Vertical Fork/Join” icon 3424 or a “Horizontal Fork/Join” activity may be used to connect more than one predecessor activities to a successor activity, or to connect a predecessor activity to more than one successor activities.

[0139] In the workflow 600 depicted in FIG. 6, the activity ID 3358 of the “Get Parts” activity 606 is “10.” The predecessor 3360 to the “Get Parts” activity 606 has an ID of “11”3362, which corresponds to the start element 602. The successor 3364 to the “Get Parts” activity 606 has an ID of “1522”3366, which corresponds to the “L or Rt Handed?” logic activity 608. The predecessor 3368 to the “L or Rt Handed?” logic activity 608 has an ID of “10”3358, which corresponds to the “Get Parts” activity 606. Because the “L or Rt Handed?” activity 608 is a logic activity, it has both a default successor and a non-default successor. Thus, the workflow definition file 3300 identifies two paths out of the “L or Rt Handed?” logic activity 608, one path 3370 has an ID of “1525”3372, which corresponds to the “Right” activity 610, and the other path 3374 has an ID of “1523”3376, which corresponds to the “Left” activity 612. The element representing the “L or Rt Handed?” logic activity 608 also identifies that the default path 3378 has an ID of “1525”3372, which corresponds to the “Right” activity 610. The predecessor 3380 to the “Right” activity 610 and the predecessor 3382 to the “Left” activity 612 have an ID of “1522”3366, which corresponds to the “L or Rt Handed?” logic activity 608. The remaining predecessor and successors follow this convention.

[0140] After checking for any successor activities and/or receiving the successor activities, the tool 200 determines whether the activity has any on-entry scripts (step 2836). An on-entry script is a step to be performed by the tool 200 upon entry into the activity. For example, the on-entry script may send an email notifying an interested user about the activity being started. The on-entry script may also send a dialog box to an enterprise affiliate to obtain data in real-time, or send a request to a separate device to gather input, e.g., by sending a message to a computer to receive data files. Other examples of on-entry scripts include checking stock levels and issuing reorder commands, if necessary, or paging the

user assigned to perform the activity. If the activity has an on-entry script, the tool **200** receives an indication of the on-entry scripts (step **2838**). After checking for any on-entry scripts and/or receiving the on-entry scripts, the tool **200** determines whether the activity has any on-exit scripts (step **2840** in **FIG. 28C**). An on-exit script is a step to be performed by the tool **200** upon exiting the activity. For example, the on-exit script may send an email notifying an interested user about the end of an activity. Other examples of on-exit scripts include sending a message to another device to have the other device perform enterprise application integration, notifying a downstream consumer about the activity so that the consumer knows what is coming, and placing an activity on a user's personal calendar. If the activity has an on-exit script, the tool **200** receives an indication of the on-exit scripts (step **2842**). For example, the "Complete Assembly" activity **620** depicted in **FIG. 6** includes both an on-entry script **3384** as well as an on-exit script **3386**. Upon entering the task created from the "Complete Assembly" activity, the tool **200** sends an email to the owner indicating that the "Debugging period started" **3388**. Prior to exiting the task created from the "Complete Assembly" activity, the tool **200** sends an email to the owner indicating that the "Debugging finished" **3390**.

[**0141**] After checking for any on-exit scripts and/or receiving the on-exit scripts, the tool **200** determines whether the activity has any input (i.e., begin or starting) conditions (step **2844**). If the activity has an input condition, the tool **200** receives an indication of the input conditions (step **2846**). Example input conditions are to expect an artifact required for the task to have a specific status. After checking for any input conditions and/or receiving the input conditions, the tool **200** determines whether the activity has any output (i.e., exit or ending) conditions (step **2848**). An example exit condition could be to automatically check the quality of an artifact generated by the task. If the artifact meets quality standards, the task completion occurs; otherwise, the task completion is rejected and the user is informed that more quality is required. If the activity has an output condition, the tool **200** receives an indication of the output conditions (step **2850**). The output condition **3391** for the "Get Parts" activity **606** has an ID of "1527" **3392** (**FIG. 33B**), and is a document-type condition, as indicated by the "linkable1" identity **3393** in the element **3394** representing the condition **3391**. In general, based on the condition **3391**, the tool **200** (in particular, the Workflow Engine **222**) monitors the state of an artifact for an activated "Get Parts" task created from the "Get Parts" activity **606** until the state of the artifact is the "INITIAL" state **3395** before the tool **200** continues with the next task in the plan. Similarly, the output condition **3396** for the "Right" activity **610** has an ID of "1533" **3397**. The output condition **3396** for the "Right" activity **610** is also a document-type condition, as indicated by the "linkable1" identity **3398**. This condition **3396** signals the tool **200** to monitor the state of an artifact until it is in the "RIGHT" state **3399**.

[**0142**] **FIG. 36** depicts an exemplary user interface **3600** displayed by the Client Interface **134** to include either a document-oriented **3602** or a script (or logic)-oriented **3604** condition. As shown in **FIG. 36**, the Client Interface **134** may receive the request to add a condition to the activity via a pull-down menu selection **3606**. The enterprise affiliate may, however, use any known data input technique to request that a condition be added to an activity, such as an

icon or keyboard input, to indicate the request to the Client Interface **134**. If the enterprise affiliate selects a document-oriented condition, the enterprise affiliate may be presented with the user interface **3700** depicted in **FIG. 37** to identify the properties of the condition to the Client Interface **134**. The condition properties **3702** include condition-name property **3704** for the document-type condition model. In the example shown in **FIG. 37**, the Client Interface **134** receives the condition-name property **3704** via a keyboard input by the enterprise affiliate. The Client Interface **134** uses the condition-name property **3704** to distinguish the condition model to be created from other condition models stored on WebDAV Storage **142**. The Client Interface **134** may store the document-type condition model file on WebDAV Storage **142** having the same name as the condition-name property **3704**. In another implementation, the Client Interface **134** may store the condition-name property **3704** as a WebDAV property stored in association with the document-type condition model file on WebDAV Storage **142**.

[**0143**] The Client Interface **134** also receives a link-parameter property **3706** as one of Condition properties **3702** for the document-type condition model to be created by the Client Interface. As shown in **FIG. 37**, the enterprise affiliate may identify link-parameter property **3706** to the Client Interface via keyboard input. Link-parameter property **3706** may be used by an enterprise affiliate in an activity-related script that is identified to the Client Interface during the creation of a workflow as described below. Thus, when executing the activity-related script in a task of a plan created from the workflow, the Workflow Engine **222** in **FIG. 2** is able to locate the corresponding document condition so that the corresponding input or output condition may be evaluated by the Workflow Engine **222**.

[**0144**] The Client Interface **134** may also receive a description property **3708** as one of Condition properties **3702** for the document-type condition model to be created by the Client Interface. When creating a workflow as described below, the Client Interface may display description property **3708** in association with condition-name property **3704** to allow an enterprise affiliate to effectively choose whether the document-type condition model should be assigned to an activity of the workflow.

[**0145**] The Client Interface may also receive one or more triggering-event properties **3710** for the document-type condition model. In the example shown in **FIG. 37**, the Client Interface may receive the triggering-event properties as one of the condition properties **3702** for the document-type condition model to be created by the Client Interface. Triggering-event properties **3710** indicate to the Workflow Engine **222** when to check the state property of a document condition as an entry or exit condition of an activated task. Triggering-event properties **3710** may include a "Write into document" event **3712**, a "Change property of document" event **3714**, a "Put document into repository" event **3716**, a "copy or move into document" event **3718**, and a "delete document" event **3720**.

[**0146**] Next, the Client Interface **134** receives document state properties **3722** as one of the Condition properties **3702** for the document-type condition model to be created by the Client Interface. Document state properties **3722** identify possible values for a state property of a document condition that is created using the document-type condition model. As

further explained herein, an enterprise affiliate who has been identified as the responsible owner of an activated task may change the state property of a document condition (e.g., from “DRAFT” to “APPROVED”) using the Client Interface, which sends a request to WebDAV Server 140 in FIG. 2 to set the state property of the document condition as indicated by the enterprise affiliate. Workflow Engine 222 in FIG. 2 may then check the state property of the document condition on WebDAV Storage 142 when triggering-events 3710 occur.

[0147] The Client Interface also receives a location property 3724 as one of Condition properties 3702 identified by the enterprise affiliate for the document-type condition model. Location property 3724 is a unique identifier or URL for a document template that the Client Interface uses to create the document condition that is then stored by the Client Interface on WebDAV Storage 142. Location property 3724 may be a location on secondary storage device 116 of computer 102a or a location on WebDAV Storage 142. As described in greater detail below, the document condition is created by the Client Interface 134 when a plan is instantiated or created from a workflow having an activity with an entry or exit condition created using the document-type condition model. Finally, the Client Interface receives application property 3726 as one of Condition properties 3702 identified by the enterprise affiliate for the document-type condition model. Application property 3726 is a unique identifier or URL for an application, such as Microsoft Word, that the Client Interface may run to create an instant of the document template that may be found at the location specified by location property 3724. The Client Interface uses the instant of the document template to create and store the document condition on WebDAV Storage 142.

[0148] FIG. 38 depicts an exemplary user interface 3800 displayed by the Client Interface 134 to receive the condition properties 3802 for a logic-type condition model that is to be created by the Client Interface 134. The condition properties 3802 include a condition-name property 3804 for the document-type condition model. In the example shown in FIG. 38, the Client Interface 134 receives the condition-name property 3804 via a keyboard input by the enterprise affiliate. The Client Interface 134 uses the condition-name property 3804 to distinguish the logic-type condition model to be created from other condition models stored on WebDAV Storage 142. As described below, the Client Interface 134 stores a logic-type condition model file on WebDAV Storage 142 that has the same name as condition-name property 3804. In another implementation, the Client Interface 134 may also store condition-name property 3804 as a WebDAV property stored in association with the logic-type condition model file on WebDAV Storage 142.

[0149] In the example shown in FIG. 38, the Client Interface 134 may receive a description property 3806 as one of the Condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. When creating a workflow as described below, the Client Interface 134 may display the description property 3806 in association with the condition-name property 3804 to allow an enterprise affiliate to effectively choose whether the logic-type condition model should be assigned to an activity of the workflow.

[0150] The Client Interface 134 may also receive one or more triggering-event properties 3808 for the logic-type

condition model as one of the condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. Triggering-event properties 3808 indicate to the Workflow Engine 222 when to check an entry or exit condition of an activated task. Triggering-event properties 3808 include: an “Absolute time” event 3810, a “Period” event 3812, a “URL change” event 3814, a “Task change” event 3816, and “any http request” event 3818. “Absolute time” event 3810 identifies a trigger for a specific data and time from the start time of the activated task. “Period” event 3812 identifies a trigger for a specific unit of time, such as once every minute. “URL change” event 3814 identifies a trigger when the contents of the directory or folder located at the URL changes. “Task change” event 3816 identifies a trigger for any time the activated task definition file or associated property changes. For example, when an enterprise affiliate that is responsible for the task uses the Client Interface 134 to identify that the task is complete, the Client Interface 134 in response sends a request to the WebDAV Server 140 to set the status property of the activated task to “FINISHED.” As part of the processing for managing an activated plan as described below, the Workflow Engine 222 will receive this request before the WebDAV Server 140 and interpret the request as an example of a “Task change” event 3816. Similarly, “Any http request” event 3818 indicates to the Workflow Engine 222 to check the entry or exit condition of the activated task when any request is received from the Client Interface 134 that pertains to the activated task. For example, the Client Interface 134 may send a request to the WebDAV Server 140 to retrieve the activated task file so that a status of the activated task can be viewed by an enterprise affiliate. Workflow Engine 222 will receive this request before the WebDAV Server 140 and interpret the request as an example of an “Any http request” event 3818.

[0151] The Client Interface 134 may also receive a script 3820 as one of the condition properties 3802 for the logic-type condition model to be created by the Client Interface 134. Script 3820 is executed by the Workflow Engine 222 when a triggering-event occurs that corresponds to one of the triggering-event properties 3808 selected by the enterprise user using the Client Interface 134. As shown in FIG. 38, Script 3820 may include a script parameter 3822, a script value 3824 for script parameter 3822, and script content 3826 that may use the script parameter 3822 initialized to the script value 3824. The enterprise affiliate may provide the script content 3826 to the Client Interface 134 via a Script Editor User Interface 3900 in FIG. 39. Script Editor User Interface 3900 is displayed by the Client Interface 134 when the enterprise affiliate actuates button 3828 on user interface 3800 shown in FIG. 38. Script content 3820 may contain any known application program interface (API) script method that would be recognizable by the target processor interpreter on computer 106, such as Java™ Virtual Machine 150 in FIG. 1.

[0152] After checking for any output conditions and/or receiving the output conditions, the tool 200 determines whether there are any more activities to add to the workflow (step 2852). If there are more activities, the process continues at step 2822 for the next activity. If there are no more activities to add to the workflow, the tool 200 receives an indication of the starting point for the workflow (step 2854). Next, the tool 200 receives an indication of the ending point for the workflow (step 2856) before the process ends.

[0153] FIG. 40 depicts an exemplary user interface 4000 displayed by the Client Interface 134 to receive the properties of an activity of a workflow. As depicted, the name 4002 of the activity (e.g., “Specs Development”), the duration 4004 of the activity (e.g., 1 unit) and the role 4006 responsible for the activity may be entered by the enterprise affiliate responsible for creating or modifying the workflow. In addition, the enterprise affiliate may enter an on-entry script 4008 as well as an on-exit script 4010. If the activity represents an entire other workflow, the properties of the activity also include the location 4012 of the sub-process defining the workflow. This allows an enterprise to save significant resources by providing a mechanism for reusing workflows within other workflows. Thus, workflows may be modularly built from constituent workflows. For example, the defect tracking workflow depicted in FIG. 34 can be used inside many “outer” or “higher-level” processes for software development.

Creating A Plan From A Workflow

[0154] FIGS. 41A-B depict a flow diagram illustrating the process of creating a plan from a workflow, i.e., step 306 in FIG. 3. At this point, the enterprise affiliate has already selected the workflow that will be used to create the plan. Initially, the tool 200 receives an indication of the plan name (step 4102). In selecting the plan name, the Client Interface 134 allows the enterprise affiliate to store the project plan within an identified project plan group so that any enterprise affiliate using the Client Interface 134 is able to easily identify related project plans. A process plan group is a collection of project plans (e.g., a directory or folder containing the collection of project plans) created by the Client Interface 134 on WebDAV Storage 142. For example, the software-related project plans may be stored within the same project plan group so that an enterprise affiliate is able to quickly locate a desired project plan in order to create a corresponding plan using the Client Interface 134. One skilled in the art will appreciate that Client Interface 134 may store a project plan on WebDAV Storage 142 without associating the project plan with a project plan group. FIG. 42 depicts an exemplary user interface 4200 used to receive a project plan group.

[0155] In the implementation shown in FIG. 42, the Client Interface 134 receives a dialog box 4202 to enter the name of a new project plan group 4204 (e.g., “Software Projects”) after clicking on a “Create Group” button 4206. Alternatively, if the enterprise affiliate decides to select an existing project plan group, the tool 200 provides the enterprise affiliate with a list 4300 of available project groups from which the enterprise affiliate may choose, as depicted in FIG. 43. The tool 200 then provides the enterprise affiliate with a dialog box 4400 to enter the name 4402 of the project, as shown in FIG. 44.

[0156] The next step performed by the tool 200 is to receive an indication of the working hours (step 4104). FIG. 45 depicts an exemplary timetable 4500 which the enterprise affiliate may use to identify the timetable defining a workday. As shown, the enterprise affiliate may select a timetable template 4502 with predefined working hours. The Standard Timetable 4504 includes five Working Days 4506 (Monday through Friday) and Working Hours 4508 from 8 a.m. (4510) through 12 p.m. (4512) and from 1 p.m. (4514) until 5 p.m. (4516). Alternatively, the enterprise affiliate may

select a 24 Hour Timetable 4518 or an Intensive Timetable 4520, i.e., more than the Standard Timetable 4504, but less than the 24 Hour Timetable 4518. The tool 200 also receives an indication of the start date and time for the project plan (step 4106). An exemplary dialog box 4600 may be used to select the start date and time 4602 and end date and time 4604.

[0157] The tool 200 then retrieves an activity from the workflow (step 4108). The tool 200 sets the start time of the task equal to the start date and time of the project plan (step 4110). Next, the tool 200 sets the end time of the task based on the start time of the task, the duration of the activity from which the task is based, and on the working hours (step 4112 in FIG. 41B). The tool 200 then receives an indication of the resource assigned to the task (step 4114).

[0158] For example, FIG. 47 depicts an exemplary workflow definition file 4700 that is produced by the tool 200 when the workflow 500 depicted in FIG. 5 is created. FIG. 48 depicts an exemplary project plan definition file 4800 created from the workflow definition file 4700. The element 4702 in the workflow definition file 4700 represents the “Serial 1” activity 506. As shown, the “Serial 1” activity 506 has a duration 4704 of 9 hours. If the working hours are determined based on the “24 Hour Timetable” 4818 and the start date and time for the project plan is 9 a.m. on Aug. 1, 2001, the start time 4804 for the “Serial 1” task 4802 is 9 a.m. on Aug. 1, 2001. The end time 4806 of the task 4802 occurs 9 hours later, i.e., at 6 p.m. on Aug. 1, 2001.

[0159] FIG. 49 depicts an exemplary user interface 4900 displayed by the Client Interface 134 to assign users or resources to the project and to assign these users specific roles related to the roles required by the project. The tool 200 displays a list of available users or resources 4902 (on the left), a list of the assigned users (central), and a list of the roles 4904 (on the right) in a given workflow. In this embodiment, the enterprise affiliate is allowed to selectively add or remove available resources to the project by highlighting the resource and selecting either the “Add” button 4906 or the “Remove” button 4908, respectively. Alternatively, the enterprise affiliate may add or remove the resources to the project by selecting the “Add all” button 4910 or the “Remove all” 4912 button, respectively. For each resource, the user can selectively indicate (checkboxes) which roles the user should play. Thus, the enterprise affiliate may identify to the tool 200 resources that are capable of performing the role when assigned to a task in the plan. As discussed below, the tool 200 may automatically assign a resource to a role of a task in the plan based on the identified, capable resources for the role.

[0160] The properties of an activity may be modified using the exemplary user interface 5000 depicted in FIG. 50. The user interface 5000 displays the name 5002 of the activity, the duration 5004 assigned to the corresponding activity, the start date and time 5006 for the activity, the end date and time 5008 for the activity, the responsible role 5010 assigned to the corresponding activity, the responsible resource or user 5012 assigned to the task, the owners 5014 of the task, the priority 5016 of the task, the on-entry script 5018 of the task, and the on-exit script 5020 of the task. The responsible resource 5012 of the task is the resource with the authority to notify the tool 200 when the task is complete. The owner(s) 5014 of the task, on the other hand, are notified

when the task is started or completed, but do not have the authority to modify the tool **200** when the task is complete.

[0161] The next step performed by the tool **200** is to determine whether there are any more activities in the workflow (step **4116**). If there are no more activities, the process ends. If there are more activities, the tool **200** retrieves the next activity (step **4118**). The tool **200** then sets the start time of the task equal to the end time of the predecessor task (step **4120**). The process then continues at step **4112**.

[0162] The next activities that are retrieved by the tool **200** are "Parallel 1" **510** and "Parallel 2" **512**. Element **4706** and element **4708** in the workflow definition file **4700** represent these activities **510** and **512**. The durations **4710** and **4712** of both of these activities is 24 hours. The start time **4812** and **4814** of these tasks **4808** and **4810** is equal to the end time **4806** of the predecessor task, i.e., 6 p.m. on Aug. 1, 2001. Because the duration **4710** and **4712** of the activities **510** and **512** is 24 hours, the end times **4816** and **4818** of these tasks **4808** and **4810** occur 24 hours later, i.e., at 6 p.m. on Aug. 2, 2001. The next activity retrieved by the tool **200** is "Serial 2" **508**. The element **4714** in the workflow definition file **4700** represents this activity. The duration **4716** of the "Serial 2" activity **508** is 24 hours. The start time **4822** of the task **4820** created from the "Serial 2" activity **508** is the end time **4816** and **4818** of the predecessor task, i.e., 6 p.m. on Aug. 2, 2001. Because the duration **4716** of the "Serial 1" activity is 24 hours, the end time **4824** of the task **4820** is 6 p.m. on Aug. 3, 2001. The project plan is displayed in the Gantt chart **5100** depicted in FIG. 51. As shown, the "Serial 1" task **5102** is scheduled to execute from 9 a.m. **5104** on Aug. 1, 2001 (**5106**) through 6 p.m. **5108** on the same day. The "Parallel 1" task **5110** and the "Parallel 2" task **5112** are scheduled to execute from 6 p.m. **5108** on Aug. 1, 2001 (**5106**) through 6 p.m. **5114** on Aug. 2, 2001 (**5116**). Finally, the "Serial 1" task **5118** is scheduled to execute from 6 p.m. **5114** on Aug. 2, 2001 (**5116**) through 6 p.m. **5120** on Aug. 3, 2001 (**5122**). Note that an enterprise affiliate using the Client Interface **134** on the computer **102a** may create a plan from the workflow **600** at the same time that a second enterprise affiliate using the Client Interface **134** on computer **102n** creates a second plan from the workflow **600**.

[0163] After the project plan is created from the workflow, the plan may be activated. As depicted in FIG. 52, the enterprise affiliate may activate the project by selecting the "Activate Project" option **5202** from the pull-down menu **5200**. The enterprise affiliate may, however, use any known data input technique, such as an icon or keyboard input, to indicate the request to Client Interface **134**.

[0164] In one implementation, the Client Interface **134** then sends an activate request to the WebDAV server **140** to change the status of the plan definition file to "Active." As discussed further below, the Workflow Engine **222** may intercept this request and process the request in preparation for managing the execution of the activated plan. Once the plan is created and stored on WebDAV storage **142**, any enterprise affiliate with appropriate privileges (e.g., project manager that "owns" the plan) may activate the plan using the Client Interface **134** from any computer **102a** and **102n**.

Adding A Resource

[0165] FIG. 53 depicts a flow diagram illustrating an exemplary process performed by the Client Interface **134** to

add a new resource to the list of available resources. The Client Interface **134** may later assign the resource to a plan in accordance with methods and systems consistent with the present invention. Initially, the Client Interface **134** receives a request to add a new resource (step **5302**). As shown in FIG. 54, the Client Interface **134** may receive the request to add a new resource via a pull-down menu selection **5402** and **5404** that is chosen by an enterprise affiliate. The enterprise affiliate may, however, use any known data input technique, such as an icon or keyboard input, to indicate the request to the Client Interface **134**.

[0166] Next, the Client Interface **134** determines whether the request is to import the resource information (step **5304**). In the implementation shown in FIG. 54, an enterprise affiliate requests that the Client Interface **134** import a resource profile containing the resource information by choosing the pull-down menu selection **5404**. Alternatively, the enterprise affiliate may request that the Client Interface **134** create the resource profile from resource information that the enterprise affiliate provides to the Client Interface **134**. Thus, if the request is not to import the resource information, the Client Interface **134** receives the resource information from the enterprise affiliate (step **5306**). As shown in FIG. 54, the Client Interface **134** may receive resource information **5404** for an enterprise affiliate (e.g., a user or person) that may later be assigned to a plan by the Client Interface **134** in accordance with processes described in greater detail below. The Resource Information **5404** may include a login name **5408**, a resource name **5410** that the Client Interface **134** is to use when assigning the resource to a task of a plan, and an e-mail address **5412** that the Client Interface **134** or the Workflow Engine **222** may use to notify the resource of an assignment or another event.

[0167] The Client Interface **134** may also receive other resource information (not shown) for other types of resources (e.g., equipment, facilities, computer systems, or other known entities) that may be assigned to any task of a plan. The other resource information may include: a resource name that the Client Interface **134** is to use when assigning the resource to a task of a plan; a resource owner name that identifies a manager or other enterprise affiliate who is responsible for the named resource; and an e-mail address for the named resource owner, which the Client Interface **134** or the Workflow Engine **222** may notify when the named resource is assigned to a task or for another associated event.

[0168] Resource information **5404** may also include one or more skill identifiers that indicate one or more capabilities that a task of a plan may require for the task to be completed. Skill identifiers may include any foreseeable skill for the named resource, including a user, equipment, facilities, computer systems, or other known entities that may be assigned to any task of a plan. For example, when the named resource is an enterprise affiliate, the skill identifiers that may be identified for the enterprise affiliate may include: "Java programming," "architecture," or "carpentry." When the named resource is equipment, the skill identifiers may include "punch-press," "printing," or "Windows NT Operating System." Or, when the resource is another system, skills may involve the ability to execute specific functions (much like distributed or web services, "credit card validation," "shop for best air freight shipper prices").

Resource information **5404** may also include a skill strength (not shown) for each skill identifier. The skill strength may be used by the tool to differentiate one resource from another resource when matching a resource to a role of a task in a plan.

[**0169**] Resource information **5404** may also include an availability timetable (not shown) that indicates to the Client Interface **134** the calendar days, the hours in a weekday, and the hours in a weekend day that the named resource is available to work. Resource information **5404** may also include an assignment timetable (not shown) that has assigned calendar days. The assigned calendar days indicate to the Client Interface **134** which calendar days the named resource has been assigned to one or more tasks. In addition, the assignment timetable may include unique identifiers or URLs for the one or more tasks to which the named resource has been assigned. Thus, the Client Interface **134** or the Workflow Engine **222** may access the one or more tasks that the named resource has been assigned when performing processing for resource leveling of a plan in accordance with methods and systems consistent with the present invention.

[**0170**] If the request is to import the resource information, the Client Interface **134** receives access information for a “Lightweight Directory Access Protocol (LDAP)” resource directory entry (e.g., a resource profile) on the network **108** of **FIG. 1** (step **5308**). **FIG. 55** depicts an exemplary user interface **5500** showing access information **5502** received by the Client Interface **134**. Access information **5502** includes an LDAP Server **5504** (e.g., “Frodo”) on the network **108**, an LDAP Port **5506** for the Client Interface **134** to communicate with the LDAP Server **5504**, and a resource distinguished name (DN) **5508** identifying the location on LDAP Server **5504** where the resource profile may be found. The access information **5502** may be default access information that the Client Interface **134** retrieves from a configuration file (not shown) on the computer **102a**, or it may be access information entered by an enterprise affiliate. In the implementation illustrated in **FIG. 55**, the access information **5502** may also include: a security distinguished name (DN) **5510**, a password **5512**, and a login alias **5514**. Security DN **5510** identifies to the Client Interface **134** where a security profile for the enterprise affiliate is located. The Client Interface **134** uses the password **5512** and the login alias **5514** to access the resource information on the LDAP Server **5504** in accordance with privileges identified in the security profile.

[**0171**] Having received the access information for the LDAP directory entry on network **108**, the Client Interface **134** retrieves the resource information using the LDAP access information (step **5310**). The resource information that the Client Interface **134** retrieves includes resource profiles for a user, equipment, facilities, computer systems, or other known entities that may be assigned to any task of a plan.

[**0172**] After the resource information is received from the enterprise affiliate or is retrieved using LDAP access information, the Client Interface **134** stores the resource information in resource profiles on the WebDAV Storage **142** (step **5312**).

[**0173**] **FIG. 56** depicts an exemplary resource file **5600** that the Client Interface **134** may use to store resource profiles **5602**, **5604**, **5606**, and **5608** on WebDAV Storage

142. As shown in **FIG. 56**, the resource profile **5600** includes a unique identifier or URL **5612** where the resource profile **5600** is to be stored on the WebDAV Storage **142**. Each resource profile **5602**, **5604**, **5606**, and **5608** may be stored separately by the Client Interface **134** on WebDAV Storage **142**. In the implementation shown in **FIG. 56**, the resource profile **5602** includes resource information **5610** that corresponds to an enterprise affiliate that may be assigned to a task of a plan. In another implementation, the resource information **5610** may be added as properties rather than as the content of the resource profile **5602** on WebDAV Storage **142**. This implementation may be advantageous as the Client Interface **134** or the Workflow Engine **222** may use a known WebDAV method to retrieve resource profiles from the WebDAV Storage **142** that have the same property. For example, the WebDAV “PropFind” method may be used by the Client Interface **134** or the Workflow Engine **222** to retrieve the resource profiles having a skill identifier of “Java Programming” so that an available resource having this skill can be assigned to a task in accordance with processes described below.

Managing A Plan

[**0174**] **FIG. 57** depicts a flow diagram illustrating an exemplary process performed by the Workflow Engine **222** to manage the execution of an activated plan. The Workflow Engine **222** may execute the process in **FIG. 57** for each activated plan stored on WebDAV Storage **142**. Thus, the tool manages the execution of multiple plans simultaneously.

[**0175**] Initially, the tool **200** waits until the current time and date are later than the start time and date (step **5702**) of the plan. Alternatively, a plan may not require a start time and date for each plan. Rather, the start time and date may be incorporated as an input condition for each task. At this point, the tool **200** selects the current next task (or tasks in the event of parallel tasks) from the activated project plan created from a workflow (step **5704**). Note that the Workflow Engine **222** may retrieve the plan from WebDAV storage. Next, the tool **200** determines whether there is an input condition (step **5706**). If there is an input condition, the tool **200** waits to see if the triggering event (described above) is met before it checks to see if the input condition is met (step **5708**). If the input condition required monitoring of certain items on a periodic basis, the Workflow Engine **222** will add this event to its “Event Monitoring” log. After the input condition is met or if there is no input condition, the tool **200** stores the actual start time (step **5710**). The next step performed by the tool **200** is to determine whether there is an on-entry script to execute, such as a message to send to the resource (step **5712** in **FIG. 57B**). If there is an on-entry script, the tool **200** performs the on-entry script (step **5714**). After performing the on-entry script or if there is no on-entry script, the tool **200** determines whether there is an output condition (step **5716**). If there is an output condition, the tool **200** waits to see if the triggering event (described above) is met before it checks to see if the output condition is met (step **5718**). After the output condition is met or if there is no output condition, the tool **200** determines whether there is an on-exit script (step **5720**). If there is an on-exit script, the tool **200** performs the on-exit script (step **5722**). After performing the on-exit script or if there is no on-exit script, the tool **200** stores the actual end time (step **5724**). Then the tool **200** determines whether there are any

more tasks in the project plan (step 5726). If there are no more tasks, the process ends. Otherwise, the process returns to step 5704 and selects the next task.

[0176] The plan 5800 created from the workflow 500 depicted in FIG. 5 is shown in FIG. 58. As shown in FIG. 58, “Serial 1” task 5802 is scheduled to begin at 9 a.m. 5804 on Aug. 1, 2001 (5806) and end at 6 p.m. 5808 on the same day. The parallel tasks 5810 and 5812 are scheduled to start at the completion of the “Serial 1” task 5808, and are scheduled to end at 6 p.m. 5814 on Aug. 2, 2001 (5816). The “Serial 2” task 5818 is scheduled to begin upon completion of the parallel tasks 5814 and is scheduled to end at 6 p.m. 5820 on Aug. 3, 2001 (5822). FIG. 59 depicts an exemplary project plan definition file 5900 corresponding to the plan 5800 of FIG. 58.

[0177] Upon activation, the “Serial 1” task 6002 begins execution, as depicted by the task 6004 in the Gantt chart 6000 of FIG. 60. Contrary to the plan, however, the “Serial 1” task ends earlier than planned. As depicted in FIG. 61, the actual properties 6100 of the “Serial 1” task 6102 include the actual-start-date 6104 (i.e., year-2001 month-8 day-1 hour-9) and actual-finish-date 6106 (i.e., year-2001 month-8 day-1 hour-14, i.e., 2 p.m.). The actual execution 6204 of the “Serial 1” task 6202 is shown in the Gantt chart 6200 of FIG. 62.

[0178] Because the “Serial 1” task 6202 ended earlier than planned, both the “Parallel 1” task 6206 and the “Parallel 2” task 6208 begin execution at 2 p.m. 6210 rather than waiting until their scheduled start time of 6 p.m. The earlier execution 6212 and 6214 of these tasks 6206 and 6208 is also depicted in the Gantt chart 6200. As depicted in FIG. 63, the actual properties 6300 of the “Parallel 1” task 6302 and the “Parallel 2” task 6304 include the actual-start-date 6306 (i.e., year-2001 month-8 day-1 hour-14) and actual-finish-date 6308 (i.e., year-2001 month-8 day-2 hour-0). The actual execution 6406 and 6408 of the “Parallel 1” task 6402 and the “Parallel 2” task 6404 is shown in the Gantt chart 6400 of FIG. 64. The Gantt chart 6400 also visually indicates that the start time 6410 for the tasks 6402 and 6404 was 2 p.m. on Aug. 1, 2001, while the end time 6412 for the tasks 6402 and 6404 was 12 a.m. on Aug. 2, 2001.

[0179] Finally, the execution of the “Serial 2” task 6414 begins at 12 a.m. on Aug. 2, 2001 (6412). As depicted in FIG. 65, the actual properties 6500 of the “Serial 2” task 6502 includes the actual-start-date 6504 (i.e., year-2001 month-8 day-2 hour-0) and actual-finish-date 6506 (i.e., year-2001 month-8 day-2 hour-12). The actual execution 6604 of the “Serial 1” task 6602, the actual execution 6608 of the “Parallel 1” task 6606, the actual execution 6612 of the “Parallel 2” task 6610, and the actual execution 6616 of the “Serial 2” task 6614, are shown in the Gantt chart 6600 of FIG. 66.

Animation Of Workflows And Project Plans

[0180] Methods and systems consistent with the present invention allow a user to animate the edits to a plan or workflow. Thus, an enterprise affiliate may view the changes made to a plan or workflow over time, or may view the various plans created from a given workflow over time. An enterprise affiliate may also use the tool 200 to review the steps performed during the activation of a plan.

[0181] FIG. 67 depicts a flow diagram illustrating an exemplary process for storing the edits to a plan definition file and the corresponding task definition files during the activation of the plan. Initially, the tool 200 retrieves a plan selected by an enterprise affiliate (step 6702). Next, the tool 200 activates the plan (step 6704). The next step performed by the tool 200 in activating the plan is to select a task from the plan (step 6706). Then, the tool 200 activates the task (step 6708). Thus, in the plan depicted in FIG. 68, upon activation, the tool 200 selects the first task 6802. The block 6804 represents the task 6802, as defined by duration and start time, on the timeline 6800 or Gantt Chart. FIG. 69 illustrates the task definition file 6900 corresponding to the task 6802 of FIG. 68. As shown in the task definition file 6900 of FIG. 69, prior to activation, the state of the task is “unexecuted”6902. After activating the task 6802, the tool 200 darkens of the outer borders of the block 7004 representing the task 7002, as depicted in the timeline 7000 of FIG. 70. The activation of the task is reflected in the task definition file (step 6710). As shown in the task definition file 7100 depicted in FIG. 71, the state of the task is changed to “executing”7102 after the task is activated. Alternatively, the state of the task may be changed to “active” rather than “executing.” The tool 200 then saves the edits to the task definition file (step 6712). The tool 200 also includes a link to the task definition file in the plan definition file, and saves the plan definition file (step 6714).

[0182] The next step performed by the tool 200 is to wait until the execution of the task is complete (step 6716). The tool 200 depicts a completed task 7202 on the timeline 7200 in FIG. 72 as a darkened block 7204. After the execution of the task is complete, the tool 200 edits the task definition file to reflect the completed task (step 6718). As shown in the task definition file 7300 of FIG. 73, the state of the task is changed to “executed”7302. After editing the task definition file, the tool 200 saves the edits to the task definition file (step 6720). The tool 200 also includes a link to the task definition file in the plan definition file, and saves the plan definition file (step 6722). Next, the tool 200 determines whether there are any more tasks (step 6724). If there are no more tasks, the process ends. Otherwise, if there are more tasks, the process continues at step 6704.

[0183] Returning to the timeline 7200 in FIG. 72, the tool 200 selects the next task 7206 from the plan, which is depicted as block 7208. The task definition file 7400 of FIG. 74 represents the second task 7206, which indicates that the state of the task is “unexecuted”7402. After activating the second task, the tool 200 darkens of the outer borders of the block 7508 representing the task 7506, as depicted in the timeline 7500 of FIG. 75. The activation of the task is reflected in the task definition file 7600 depicted in FIG. 76. In particular, the state of the task is changed to “executing”7602 after the task is activated. The tool 200 then saves the edits to the task definition file 7600, includes a link to the task definition file 7600 in the plan definition file, and saves the plan definition file. The tool 200 then waits until the execution of the task is complete, and depicts the completed task 7706 on the timeline 7700 in FIG. 77 as a darkened block 7708. After the execution of the task is complete, the tool 200 edits the task definition file 7800 of FIG. 78 to reflect the completed task, i.e., the state of the task is changed to “executed”7802. After editing the task definition file 7800, the tool 200 saves the edits to the task definition file 7800, includes a link to the task definition file 7800 in

the plan definition file, and saves the plan definition file. In the example above, the tool represents an executing task with darkened borders and represents an executed task as a darkened block. One skilled in the art, however, will recognize that any visible change in the blocks representing the tasks, e.g., a change in shape, color, shading, etc., may be used to represent the tasks in their various states. Thus, the representation of the tasks used in the methods, systems, and articles of manufacture consistent with the present invention are not limited to those used in the present embodiment.

[0184] In another implementation, the tool 200 allows a user to store the modifications made to a plan, and allows the user to view the changes made to a plan over time. FIG. 79 depicts a flow diagram illustrating an exemplary process for storing the edits to a plan. Initially, the tool 200 retrieves a plan selected by a user (step 7902). The user makes modifications to a task in the plan, which are reflected in the task definition file by the tool 200 (step 7904). Next, the tool 200 saves the edits to the task definition file (step 7906). The tool 200 also includes a link to the task definition file in the plan definition file and saves the plan definition file (step 7908). The next step performed by the tool 200 is to determine whether there are any more changes to be made to the plan (step 7910). If there are no more changes to be made, the process ends. Otherwise, if there are more changes, the process continues at step 7904. The modifications to the tasks may include changing the resource assigned to the task or changing the start time of the task. The modifications to the tasks may be made before the activation of the plan. Alternatively, modifications may be made to the tasks during the execution of the plan as long as the task that is being modified has not yet become active.

[0185] In general, the storage of the complete version history of a workflow or a plan allows methods and systems consistent with the present invention to sequentially step through the versions, display the workflow or plan, and provide "video cassette recorder"-like navigation (e.g., pause/resume, play, forward, reverse, go to start, go to end) through the versions of the workflow or plan. Forward implies going from an earlier version, forward in time, through newer versions. Going backwards implies starting from, for example, the current version and tracing back through the earlier versions; for example, to the initial version.

[0186] All available versions of a plan may be retrieved, and the user may choose the range of versions desired. The system may be set up to retrieve all versions by default. The system may also use a VCR-like mechanism to receive the indication of how the user wishes to step through the versions. The system may then receive an indication from the user as to how to view the animation (e.g., play forward from the beginning). When it is determined to display in the forward mode, the method may retrieve the earliest version in the selected range and displaying the plan. At a user-selectable rate (e.g., display one version every 5 seconds), the system may retrieve the next version, apply the edits to the plan, and display the plan. The portions of the plan may be visually distinctive as a function of frequency of change. For example, areas of the plan that do not change from version to version may remain in a visually non-distinctive color. Those areas that undergo the most change may be visually distinct (e.g., red or bolded or tagged with a number indicating changes). Those areas that are "removed" may

also be visually distinct (e.g., grayed out, or faint or tagged with a small "removed" symbol).

[0187] FIGS. 80A and B depict the process performed by the tool 200 to animate the changes to the plan. Initially, the tool 200 retrieves the edits to the plan definition file (step 8002). In one implementation, each edit may be stored in a separate file with a link to the plan definition file. In another implementation, all edits may be stored in a single file with a link to the plan definition file. In yet another implementation, all edits may be stored with the plan definition file. The tool 200 also retrieves edits to the task definition files (step 8004). Similar to the edits to the plan definition file, each edit may be stored in a separate file with a link to the task definition file. Alternatively, all of the edits may be stored in a single file with a link to the task definition file, or all of the edits may be stored with the task definition file. Then, the user sets the rate of the display (step 8006). Next, the tool 200 sets the time period equal to the reciprocal of the rate (step 8008). The time period indicates the amount of time the tool 200 pauses between the different displays of the animation. Thus, if the rate is 1/sec, the tool pauses 1 sec. between each of the different displays of the animation. The enterprise affiliate may choose to display the animation in the forward or reverse direction. Thus, the tool 200 determines whether the user chose to display the animation in the forward mode (step 8010). If the tool 200 determines that the animation will be displayed in the forward mode, the tool 200 removes the edits to the plan definition file (step 8012). The tool 200 also removes the edits to the task definition files (step 8014). Next, the tool 200 displays the plan (step 8016). The tool 200 then pauses for the time period (step 8018). After waiting the time period, the tool 200 selects the first edit (step 8020). The next step performed by the tool 200 is to apply the edit (step 8022). The tool 200 then displays the edited plan (step 8024). The tool 200 also determines whether the enterprise affiliate has decided to adjust the rate of the display (step 8026). If the tool 200 receives a request from the user to adjust the rate of the display, the tool 200 resets the time period to the reciprocal of the new rate (step 8028). Then, the tool 200 determines whether there are any more edits (step 8030). If there are no more edits, the process ends. Otherwise, if there are additional edits, the process continues at step 8018.

[0188] If the enterprise affiliate chose not to display the animation in the forward mode, the next step performed by the tool 200 is to display the plan (step 8032 in FIG. 80B). Next, the tool 200 pauses for the time period (step 8034). The tool 200 then selects an edit (step 8036). After selecting the edit, the tool 200 removes the edit (step 8038). The tool 200 then displays the edited plan (step 8040). The next step performed by the tool 200 is to determine whether the enterprise affiliate has requested an adjustment in the rate of display (step 8042). If the tool 200 determines that the enterprise affiliate requested an adjustment to the rate, the tool 200 resets the time period to the reciprocal of the new rate (step 8044). The tool 200 then determines whether there are any more edits (step 8046). If there are no more edits, the process ends. Otherwise, if there are additional edits, the process continues at step 8034.

[0189] Similar to the plan discussed above, methods and systems consistent with the present invention may be used to animate changes to a workflow. FIG. 81 depicts a flow diagram illustrating an exemplary process for storing indi-

cations of edits to a workflow. Initially, the tool 200 retrieves a workflow (step 8102). For example, an enterprise affiliate may choose the workflow 8200 depicted in FIG. 82. The workflow includes a start element 8202 and an end element 8204. The workflow depicted also includes “Get Parts” activity 8206 followed by “L or Rt Handed?” logic activity 8208. The activity definition file 8300 representing “Get Parts” activity 8206 is depicted in FIG. 83. The default path, represented by a solid line out of the decision block or logic activity 8208, leads to a “Right” activity 8214, which is followed by a “Complete Assembly” activity 8216. The activity definition file 8400 representing “Right” activity 8214 is depicted in FIG. 84, and the activity definition file 8500 representing “Complete Assembly” activity 8216 is depicted in FIG. 85. The non-default path, represented by a dashed line out of the decision block or logic activity 8208, leads to a “Left” activity 8210, followed by a “Left Special” activity 8212, and the “Complete Assembly” activity 8216. The activity definition file 8600 representing “Left” activity 8210 is depicted in FIG. 86, and the activity definition file 8700 representing “Left Special” activity 8212 is depicted in FIG. 87. In response to a modification made to the workflow, the tool 200 edits an activity in the workflow (step 8104). Thus, if the “Left Special” activity 8212 of workflow of FIG. 82 were removed, the resulting workflow 8800 is depicted in FIG. 88. Because the “Left Special” activity 8212 was between the “Left” activity 8210 and the “Complete Assembly” activity 8216, the activity definition files corresponding to these activities will be edited. The activity definition file corresponding to the revised “Left” activity 8810 is depicted in FIG. 89, and the activity definition file corresponding to the “Complete Assembly” activity 8814 is depicted in FIG. 90. The two activity definition files corresponding to the “Left” activity before and after the removal of the “Left Special” activity 8212 are depicted in FIGS. 86 and 89, respectively. In particular, the successor from the “Left” activity is changed from id 1524 (8602) to id 1526 (8902), which corresponds to a change in successor from the “Left Special” activity 8702 to the “Complete Assembly” activity 8502. Similarly, modification to the activity definition file for the “Complete Assembly” activity indicates that the predecessor activity changed from the “Left Special” activity 8504 in FIG. 85 to the “Left” activity 9002 in FIG. 90.

[0190] The next step performed by the tool 200 is to save the edits to the activity definition file (step 8106). The tool 200 also saves the edits to the workflow definition file (step 8108). The tool 200 then determines whether there are any more changes made to the workflow (step 8110). If there are no more changes, the process ends. Otherwise, the process continues at step 8104.

[0191] In another implementation, the tool 200 allows a user to store the different plans created from one workflow. FIG. 91 depicts a flow diagram illustrating an exemplary process for storing the different plans. Initially, the tool 200 creates a plan from the workflow (step 9102). The tool 200 then stores the plan definition file (step 9104). The tool 200 creates a link from the workflow definition file to the plan definition file and stores the edited workflow definition file (step 9106). Next, the tool 200 creates a different plan from the workflow (step 9108). After creating the different plan, the tool 200 stores the different plan definition file (step 9110). The next step performed by the tool 200 is to include a link to the different plan definition file with the workflow

definition file and store the edited workflow definition file (step 9112). The tool 200 then determines whether to create more plans (step 9114). If the tool 200 determines that no additional plans will be created, the process ends. Otherwise, the process continues at step 9108.

[0192] FIGS. 92A and B depict the process performed by the tool 200 to animate the changes to the workflow. Initially, the tool 200 retrieves the edits to the workflow definition file (step 9202). In one implementation, each edit may be stored in a separate file with a link to the workflow definition file. In another implementation, all edits may be stored in a single file with a link to the workflow definition file. In yet another implementation, all edits may be stored with the workflow definition file. The tool 200 also retrieves edits to the activity definition files (step 9204). Similar to the edits to the workflow definition file, each edit may be stored in a separate file with a link to the activity definition file. Alternatively, all of the edits may be stored in a single file with a link to the activity definition file, or all of the edits may be stored with the activity definition file. Then, the user sets the rate of the display (step 9206). Next, the tool 200 sets the time period equal to the reciprocal of the rate (step 9208). The time period indicates the amount of time the tool 200 pauses between the different displays of the animation. Thus, if the rate is 1/sec, the tool pauses 1 sec. between each of the different displays of the animation. Similar to the above implementation, the enterprise affiliate may choose to display the animation in the forward or reverse direction. Thus, the tool 200 determines whether the user chose to display the animation in the forward mode (step 9210). If the tool 200 determines that the animation will be displayed in the forward mode, the tool 200 removes the edits to the workflow definition file (step 9212). The tool 200 also removes the edits to the activity definition files (step 9214). Next, the tool 200 displays the workflow (step 9216). The tool 200 then pauses for the time period (step 9218). After waiting the time period, the tool 200 selects the first edit (step 9220). The next step performed by the tool 200 is to apply the edit (step 9222). The tool 200 then displays the edited workflow (step 9224). The tool 200 also determines whether the enterprise affiliate has decided to adjust the rate of the display (step 9226). If the tool 200 receives a request from the enterprise affiliate to adjust the rate, the tool 200 resets the time period to the reciprocal of the new rate (step 9228). Then, the tool 200 determines whether there are any more edits (step 9230). If there are no more edits, the process ends. Otherwise, if there are additional edits, the process continues at step 9218.

[0193] If the enterprise affiliate chose not to display the animation in the forward mode, the next step performed by the tool 200 is to display the plan (step 9232 in FIG. 92B). Next, the tool 200 pauses for the time period (step 9234). The tool 200 then selects an edit (step 9236). After selecting the edit, the tool 200 removes the edit (step 9238). The tool 200 then displays the edited workflow (step 9240). The next step performed by the tool 200 is to determine whether the enterprise affiliate has requested an adjustment in the rate of display (step 9242). If the tool 200 determines that the enterprise affiliate requested an adjustment in the rate, the tool 200 resets the time period to the reciprocal of the new rate (step 9244). The tool 200 then determines whether there are any more edits (step 9246). If there are no more edits, the process ends. Otherwise, if there are additional edits, the process continues at step 9234.

[0194] While various embodiments of the present invention have been described, it will be apparent to those of skill in the art that many more embodiments and implementations are possible that are within the scope of this invention. Accordingly, the present invention is not to be restricted except in light of the attached claims and their equivalents.

What is claimed is:

1. A method in a data processing system, comprising the steps of:

- retrieving edits to a plan;
- receiving an indication of a rate of display;
- setting a time period equal to a reciprocal of the rate;
- determining whether to display in a forward mode;
- when it is determined to display in the forward mode,
 - removing the edits from the plan;
 - displaying the plan; and
 - for each of the edits,
 - applying the edit to the plan;
 - displaying the plan;
 - pausing for the time period;
 - determining whether to adjust the rate of the display; and
 - when it is determined that the rate of the display will be adjusted,
 - receiving an indication of a new rate of display; and
 - setting the time period equal to a reciprocal of the new rate; and
- when it is determined not to display in the forward mode,
 - displaying the plan; and
 - for each of the edits,
 - removing the edit from the plan;
 - displaying the plan;
 - pausing for the time period;
 - determining whether to adjust the rate of the display; and
 - when it is determined that the rate of the display will be adjusted,
 - receiving an indication of the new rate of display; and
 - setting the time period equal to the reciprocal of the new rate.

2. A method in a data processing system, comprising the steps of:

- retrieving edits to a workflow;
- receiving an indication of a rate of display;
- setting a time period equal to a reciprocal of the rate;
- determining whether to display in a forward mode;
- when it is determined to display in the forward mode,

- removing the edits from the workflow;

- displaying the workflow; and

- for each of the edits,

- applying the edit to the workflow;

- displaying the workflow;

- pausing for the time period;

- determining whether to adjust the rate of the display; and

- when it is determined that the rate of the display will be adjusted,

- receiving an indication of a new rate of display; and

- setting the time period equal to a reciprocal of the new rate; and

- when it is determined not to display in the forward mode,

- displaying the workflow; and

- for each of the edits,

- removing the edit from the workflow;

- displaying the workflow;

- pausing for the time period;

- determining whether to adjust the rate of the display; and

- when it is determined that the rate of the display will be adjusted,

- receiving an indication of the new rate of display; and

- setting the time period equal to the reciprocal of the new rate.

3. A method in a data processing system having versions of a plan, each reflecting an instance in an edit history, the method comprising the steps of:

- storing indications of the versions of the plan; and

- displaying the versions of the plan in a sequential manner to simulate animation of the edit history.

4. The method of claim 3, wherein the step of storing indications of the versions of the plan comprises the steps of:

- storing versions of a task of the plan; and

- creating a link from the plan to the versions of the task.

5. The method of claim 3, wherein the versions of the plan reflect an activation of the plan.

6. The method of claim 5, wherein the plan comprises a plurality of tasks, and the indications of the versions of the plan comprise the states of the tasks.

7. The method of claim 6, wherein the state comprises an unexecuted state.

8. The method of claim 6, wherein the state comprises an executing state.

9. The method of claim 6, wherein the state comprises an executed state.

10. The method of claim 3, wherein the indications of versions of the plan reflect a modification to the plan.

11. The method of claim 3, wherein the versions of the plan are displayed in reverse order.

12. The method of claim 3, wherein the display comprises a Gantt chart.

13. A method in a data processing system having versions of a workflow, each reflecting an instance in an edit history, the method comprising the steps of:

storing indications of the versions of the workflow; and
displaying the versions of the workflow in a sequential manner to simulate animation of the edit history.

14. The method of claim 13, wherein the step of storing indications of the versions of the workflow comprises the steps of:

storing versions of an activity of the workflow; and
creating a link from the workflow to the versions of the activity.

15. The method of claim 13 wherein the indications of versions of the workflow reflect a modification to the workflow.

16. The method of claim 13 wherein the versions of the workflow are displayed in reverse order.

17. The method of claim 13 wherein the display comprises a flow diagram.

18. A computer-readable medium containing instructions for controlling a data processing system to perform a method, the method comprising the steps of:

retrieving edits to a plan;
determining whether to display in a forward mode;
when it is determined to display in the forward mode,
removing the edits from the plan;
displaying the plan; and
for each of the edits,
applying the edit to the plan; and
displaying the plan; and
when it is determined not to display in the forward mode,
displaying the plan; and
for each of the edits,
removing the edit from the plan; and
displaying the plan.

19. The computer-readable medium of claim 18 wherein the plan comprises a plurality of tasks and the edits to the plan reflect a state of each task in the plan.

20. The computer-readable medium of claim 19 wherein the state comprises an unexecuted state.

21. The computer-readable medium of claim 19 wherein the state comprises an executing state.

22. The computer-readable medium of claim 19 wherein the state comprises an executed state.

23. The computer-readable medium of claim 18 wherein the edits to the plan reflect a modification to the plan.

24. The computer-readable medium of claim 18 wherein the plan is displayed in a Gantt chart.

25. A computer-readable medium containing instructions for controlling a data processing system to perform a method, the method comprising the steps of:

retrieving edits to a workflow;
determining whether to display in a forward mode;

when it is determined to display in the forward mode,

removing the edits from the workflow;
displaying the workflow; and
for each of the edits,
applying the edit to the workflow; and
displaying the workflow; and

when it is determined not to display in the forward mode,
displaying the workflow; and

for each of the edits,
removing the edit from the workflow; and
displaying the workflow.

26. The computer-readable medium of claim 25 wherein the edits to the workflow reflect a modification to the workflow.

27. The computer-readable medium of claim 25 wherein the workflow is displayed in a flow diagram.

28. A computer-readable medium containing instructions for controlling a data processing system to perform a method, the method comprising the steps of:

retrieving a plurality of plans generated from a workflow;
and

displaying each of the plans in a sequential manner to simulate the generation of the plans from the workflow.

29. The computer-readable medium of claim 28, farther comprising the steps of:

receiving an indication of a rate of display;
setting a time period equal to a reciprocal of the rate; and
pausing for the time period before displaying each of the plans.

30. The computer-readable medium of claim 28, wherein the plans are displayed in reverse order.

31. The computer-readable medium of claim 28 wherein the display comprises a Gantt chart.

32. A computer-readable medium containing instructions for controlling a data processing system to perform a method, the data processing system comprising a plan and the plan comprising a plurality of tasks, the method comprising the steps of:

displaying a graphical representation of the plan, wherein the graphical representation has portions that correspond to the tasks;

retrieving edits to the plan, wherein each of the edits modifies a state of one of the plurality of tasks; and

for each of the edits,
applying the edit to the corresponding task of the plan;
and

displaying the portion of the graphical representation that corresponds to the edited task in a visually distinctive manner.

33. The computer-readable medium of claim 32, wherein the state comprises an unexecuted state.

34. The computer-readable medium of claim 32, wherein the state comprises an executing state.

35. The computer-readable medium of claim 32, wherein the state comprises an executed state.

36. The computer-readable medium of claim 32, wherein the graphical display comprises a Gantt chart.

37. The computer-readable medium of claim 32, wherein the method further comprises the steps of:

receiving an indication of a rate of display;

setting a time period equal to a reciprocal of the rate; and

pausing for the time period before applying the edit to the corresponding task of the plan for each of the edits.

38. A data processing system comprising:

a secondary storage device further comprising a workflow;

a memory device further comprising a program that retrieves the workflow from the secondary storage device, that creates versions of the workflow, wherein each version reflects an instance in an edit history, that stores the versions of the workflow on the secondary storage device, and that displays the versions of the workflow in a sequential manner to simulate animation of the edit history; and

a processor for running the program.

39. The data processing system of claim 38, wherein the program displays the versions of the workflow in reverse order.

40. The data processing system of claim 38, wherein the program displays the versions of the workflow as flow diagrams.

41. A data processing system comprising:

a secondary storage device further comprising a plan;

a memory device further comprising a program that retrieves the plan from the secondary storage device, that creates versions of the plan, wherein each version reflects an instance in an edit history, that stores the versions of the plan on the secondary storage device, and that displays the versions of the plan in a sequential manner to simulate animation of the edit history; and

a processor for running the program.

42. The data processing system of claim 41, wherein the program displays the versions of the plan in reverse order.

43. The data processing system of claim 41, wherein the program displays the versions of the plan as Gantt charts.

44. A system having versions of a plan, each reflecting an instance in an edit history, the system comprising:

means for storing indications of the versions of the plan; and

means for displaying the versions of the plan in a sequential manner to simulate animation of the edit history.

* * * * *