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

(19) **World Intellectual Property Organization**
International Bureau

(43) **International Publication Date**
19 September 2002 (19.09.2002)

PCT

(10) **International Publication Number**
WO 02/073479 A2

(51) **International Patent Classification**⁷: **G06F 17/60**

(21) **International Application Number:** PCT/EP02/00710

(22) **International Filing Date:** 24 January 2002 (24.01.2002)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**
01104202.5 22 February 2001 (22.02.2001) EP

(71) **Applicant** *(for all designated States except US)*: **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) **Applicant** *(for LU only)*: **IBM DEUTSCHLAND GMBH** [DE/DE]; Pascalstrasse 100, 70569 Stuttgart (DE).

(72) **Inventors; and**
(75) **Inventors/Applicants** *(for US only)*: **LEYMANN, Frank** [DE/DE]; Hasenäckerweg 19, 71134 Aidlingen 3 (DE). **ROLLER, Dieter** [DE/DE]; Hermann-Löns-Weg 5, 71101 Schönaich (DE).

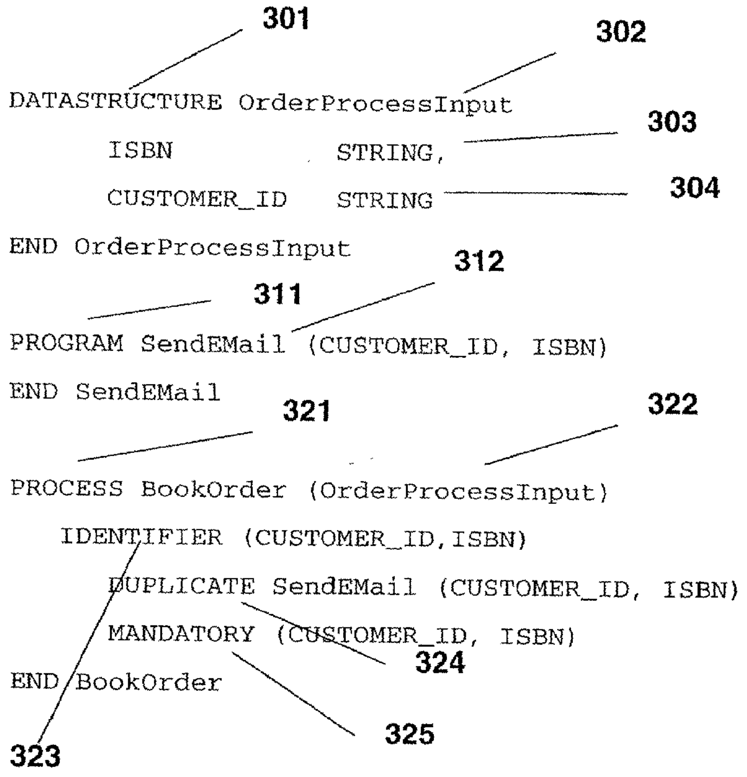
(74) **Agent: DUSCHER, Reinhard**; IBM Deutschland GmbH, Intellectual Property, Pascalstr. 100, 70548 Stuttgart (DE).

(81) **Designated States** *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),

[Continued on next page]

(54) **Title:** CONTROLLING THE CREATION OF PROCESS INSTANCES IN WORKFLOW MANAGEMENT SYSTEMS



```
DATASTRUCTURE OrderProcessInput
     ISBN           STRING,
     CUSTOMER_ID    STRING
END OrderProcessInput

PROGRAM SendEMail (CUSTOMER_ID, ISBN)
END SendEMail

PROCESS BookOrder (OrderProcessInput)
   IDENTIFIER (CUSTOMER_ID,ISBN)
      DUPLICATE SendEMail (CUSTOMER_ID, ISBN)
      MANDATORY (CUSTOMER_ID, ISBN)
END BookOrder
```

(57) **Abstract:** The present invention relates to a method and to a system for controlling the creation of process instances within an execution environment. Upon receiving a request to create a process instance and corresponding input data to be processed by said process instance, the method and corresponding system in a first step constructs a process instance identifier for said process instance to be created based on the provided input. In a second step the method and corresponding system is executing said create request only, if no process instance exists or existed with said process instance identifier. In the case, that that particular process instance already exists or existed with said process instance identifier, the present invention optionally proposes to have some user-defined action to be carried out.

WO 02/073479 A2

Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Controlling the Creation of Process Instances In Workflow Management Systems

1. Background of the Invention

1.1 Field of the Invention

The present invention relates to means and a method for improving the robustness and ease-of-use of Workflow-Management-Systems or a computer system with comparable functionality (WFMS) related to the creation of process instances from process models.

1.2 Description and Disadvantages of Prior Art

A new area of technology with increasing importance is the domain of Workflow-Management-Systems (WFMS). WFMS support the modeling and execution of business processes. Business processes executed within a WFMS environment control who will perform which piece of work of a network of pieces of work and which resources are exploited for this work. The individual pieces of work might be distributed across a multitude of different computer systems connected by some type of network.

The product "IBM MQSeries Workflow" (previously called IBM FlowMark) represents such a typical modern, sophisticated, and powerful workflow management system. It supports the modeling of business processes as a network of activities. This network of activities, the process model, is constructed as a directed, acyclic, weighted, colored graph. The nodes of the graph represent the activities, which are performed. The edges of the graph, the control connectors, describe the potential sequence of execution of the activities. Definition of the process graph is via IBM MQSeries Workflow's Flow Definition Language (FDL) or via the built-in graphical editor.

The runtime component of the Workflow-Management-System uses said process model as a template to create process instances. Each process instance is associated with a set of values, typically called the **context**. Said values are either supplied by the requestor of the process instance via the appropriate request or retrieved by programs that implement the various activities. A particular important piece of information within said context is the process instance identifier that uniquely identifies a process instance. It should be noted that typically process instance identifiers are only unique within the set of process instances that are derived from a particular process model.

The creation of a process instance is initiated by an appropriate request being made to the Workflow-Management-System. Many options exist for requestors to initiate the request; typical examples are putting a message into a queue, the sending of e-mail to a particular address, the issue of an HTTP post request to an URL, or the invocation of a function of the application-programming interface offered by the workflow management system. The requestor could be any type of program or human interacting with the system directly; it could even be the workflow management system itself. Typically the request has data (also part of the context) associated with it that allows the requestor to immediately provide information for the process instance context.

The process instance identifier is either generated by the Workflow-Management-System or supplied by the requestor of the process instance. Both state-of-the-art approaches have significant disadvantages.

When the Workflow-Management-System generates the process instance identifier, every request to create a process instance automatically results in the creation of a process instance. Thus, if the same request is carried out twice (or in general multiple times), two process instances are created. Carrying out the same request twice could happen as the result of many different situations:

- A user interacting with the Workflow-Management-System via a business application starts accidentally the same request twice. For instance clerks unconsciously repeat a certain business process for a certain set of input data multiple times not knowing or not remembering that it has been processed already.

- A request sent via unreliable e-mail is carried out twice. This situation can happen, for example, if the computer system on which the Workflow-Management-System is running and to which the e-mail was sent, crashed. In this case, it can happen that the e-mail has disappeared from one of the involved systems but not from the other; thus it is unclear whether the e-mail was processed or not. The usual solution to this situation is to reprocess the e-mail again with the undesirable result that the same business process processing the same input data multiple times.

The other state-of-the-art approach, wherein the requester needs to generate the process instance identifier, has also significant disadvantages, in particular if there is more than one requester creating process instances, which in general is the typical situation. In this case, no two requesters must use the same process instance identifier; thus requesters must share some common method to obtain unique process instance identifiers. This requires, for example, that any newly added requestor implements said common method; or in other words, the requests have to check against one another to exploit the same approach for creating process instance identifiers. This becomes very quickly an administrative and program development nightmare.

The weakness of the state-of-the-art approaches with respect to this problem area becomes even more distinct if one thinks of typical Internet scenarios commonly summarized by terms like C2B (Consumer-to-Business) or B2B (Business-to-Consumer) business processes. In these scenarios, it is obvious that neither can the requestor generate the process instance identifier nor can it be guaranteed that a request is issued only once and only once.

1.2 Objective of the Invention

The invention is based on the objective to eliminate the risk of creating duplicate process instances for business processes created and managed by a Workflow-Management-System.

2. Summary and Advantages of the Invention

The objectives of the invention are solved by the independent claims. Further advantageous arrangements and embodiments of the invention are set forth in the respective subclaims.

The present invention relates to a method and to a system for controlling the creation of process instances within an execution environment. Upon receiving a request to create a process instance and corresponding input data to be processed by said process instance, the method and corresponding system in a first step constructs a process instance identifier for said process instance to be created based on the provided input data. In a second step the method and corresponding system is executing said create request only, if no process instance exists or existed in the past with said process instance identifier. In the case, that that particular process instance already exists or existed in the past with said process instance identifier, the present invention optionally proposes to have some user-defined action to be carried out.

The suggested approach eliminates the possibility that multiple process instances are created representing the same create request. In addition, the support of an action to be taken if a process instance with the new process instance identifier exists already, provides, for example, for the notification of appropriate service personnel to take appropriate actions.

3. Brief Description of the Drawings

Figure 1 shows an example of a process model represented by a process graph.

Figure 2 illustrates on an exemplary level the states a process instance can take when it is carried out by the Workflow-Management-System.

Figure 3 visualizes the details of the specification mechanism for process instance identifiers using the Flow Definition Language of MQSeries Workflow, which form the basis of controlling the generation of process instances according to the current invention.

4. Description of the Preferred Embodiment

In the drawings and specification there has been set forth a preferred embodiment of the invention and, although specific terms are used, the description thus given uses terminology in a generic and descriptive sense only and not for purposes of limitation. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims.

The present invention can be realized in hardware, software, or a combination of hardware and software. Any kind of computer system - or other apparatus adapted for carrying out the methods described herein - is suited. A typical combination of hardware and software could be a general-purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein. The present invention can also be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which - when being loaded in a computer system - is able to carry out these methods.

Computer program means or computer program in the present context mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following a) conversion to another language, code or notation; b) reproduction in a different material form.

The current invention is illustrated based on IBM's "MQSeries Workflow" workflow management system. Of course any other WFMS could be used instead. Furthermore the current teaching applies also to any other type of system, which offers WFMS functionalities not as a separate WFMS but within some other type of system.

Though the created requests of the following examples are processed by the WFMS engine this should not be understood as a limitation. The current invention can be applied in other scenarios wherein the processing entity of the create command is not the WFMS engine itself.

4.1 Introduction

The following is a short outline on the basic concepts of a workflow management system based on IBM's "MQSeries Workflow" WFMS:

From an enterprise point of view the management of business processes is becoming increasingly important: **business processes** or **process** for short control which piece of work will be performed by whom and which resources are exploited for this work, i.e. a business process describes how an enterprise will achieve its business goals. A WFMS may support both, the modeling of business processes and their execution.

Modeling of a business process as a syntactical unit in a way that is directly supported by a software system is extremely desirable. Moreover, the software system can also work as an interpreter basically getting as input such a model: The model, called a **process model** or **workflow model**, can then be instantiated and the individual sequence of work steps depending on the context of the instantiation of the model can be determined. Such a model of a business process can be perceived as a template for a class of similar processes performed within an enterprise; it is a schema describing all possible execution variants of a particular kind of business process. An instance of such a model and its interpretation represents an individual process, i.e. a concrete, context dependent execution of a variant prescribed by the model. A WFMS facilitates the management of business processes. It provides a means to describe models of business processes (buildtime) and it drives business processes based on an associated model (runtime). The meta model of IBM's WFMS MQSeries Workflow, i.e. the syntactical elements provided for describing business process models, and the meaning and interpretation of these syntactical elements, is described next.

A process model is a complete representation of a process, comprising a process diagram and the settings that define the logic behind the components of the diagram. Important components of a MQSeries Workflow process model are:

- Processes
- Activities
- Blocks
- Control Flows
- Connectors
- Data Containers
- Data Structures
- Conditions
- Programs
- Staff

Not all of these elements will be described below.

Activities are the fundamental elements of the meta model. An activity represents a business action that is from a certain perspective a semantic entity of its own.

A MQSeries Workflow process model consists of the following types of activities:

Program activity: Has a program assigned to perform it. The program is invoked when the activity is started. In a fully automated workflow, the program performs the activity without human intervention. Otherwise, the user must start the activity by selecting it from a runtime work list. Output from the program can be used in the exit condition for the program activity and for the transition conditions to other activities.
Process activity: Has a (sub-)process assigned to perform it. The process is invoked when the activity is started. A process activity represents a way to reuse a set of activities that are common to different processes. Output from the process, can be used in the exit condition for the process activity and for the transition conditions to other activities.

The flow of control, i.e. the **control flow** through a running process determines the sequence in which activities are executed. The MQSeries Workflow workflow manager navigates a path through the process that is determined by the evaluation to TRUE of start conditions, exit conditions, and transition conditions.

Connectors link activities in a process model. Using connectors, one defines the sequence of activities and the transmission of data between activities. Since activities might not be executed arbitrarily they are bound together via **control connectors**. A control connector might be perceived as a directed edge between two activities; the activity at the connector's end point cannot start before the activity at the start point of the connector has finished (successfully). Control connectors model thus the potential flow of control within a business process model. Default connectors specify where control should flow when the transition condition of no other control connector leaving an activity evaluates to TRUE. Default connectors enable the workflow model to cope with exceptional events. Data connectors specify the flow of data in a workflow model. A data connector originates from an activity or a block, and has an activity or a block as its target. One can specify that output data is to go to one target or to multiple targets. A target can have more than one incoming data connector.

Process definition includes modeling of activities, control connectors between the activities, input/output container, and data connectors. A process is represented as a directed acyclic graph with the activities as nodes and the control/data connectors as the edges of the graph. The graph is manipulated via a built-in graphic editor. The data containers are specified as named data structures. These data structures themselves are specified via the DataStructureDefinition facility. Program activities are implemented through programs. The programs are registered via the Program Definition facility. Blocks contain the same constructs as processes, such as activities, control connectors etc. They are however not named and have their own exit condition. If the exit condition is not met, the block is started again. The block thus implements a Do Until construct. Process activities are implemented as processes. These subprocesses are defined separately as regular, named processes with all its usual properties. Process activities offer great flexibility for process definition. It not only allows to construct a process through permanent refinement of activities into program and process activities (top-down), but also to build a process out of a set of existing processes (bottom-up).

All programs, which implement program activities, are defined via the Program Registration Facility. Registered for each program is the name of the program, its location, and the invocation string. The invocation string consists of the program name and the command string passed to the program.

As an example of such a process model Fig. 1 shows schematically the structure of such a process graph. Activities (A1 up to A5) are represented as named circles; the name typically describes the purpose of the activity. Activities come in various flavors to address the different tasks that may need to be performed. They may have different activity implementations to meet these diverse needs. **Program activities** are performed by an assigned program, **process activities** like for instance 100 are performed by another process 101, and **blocks** like for instance 102 implement a macro 103 with a built-in do-until loop.
Control connectors p12, p13, p24, p35, p45 are represented as arrows; the head of the arrow describes the direction in which the flow of control is moving through the process. The activity where the control connector starts is called the **source activity**; where it ends is called the **target activity.** When more than one control connector leaves an activity, this indicates potentially parallel work.

4.2 Process States

A process instance occupies various states when it is carried out by the Workflow-Management-System. Fig. 2 illustrates those states exemplary. It should be noted that this for illustration purpose only; Workflow-Management-Systems typically differentiate between many more states.

The first step a particular process instance goes through is that it is created by taking the appropriate process template (process model), possibly populating it with supplied context data, and assigning it a unique process instance identifier. This step is carried out as the result of invoking the

Workflow-Management-System's CREATE function. As a result of function completion, the process instance is put into the state **created** 201.

When the process instance is being carried out, that means the Workflow-Management-System navigates through the process graph and executes the individual activities, the process is in the state **running** 202. The business process is typically put into this state by a client issuing a START control command; other possibilities are that the business process is automatically started by the Workflow-Management-System at a time specified when the business process is created, or a combination of a CREATE and START control command.

When all necessary activities of the process instance have been carried out, the process goes into the state **finished** 203. No further activities are carried out for the process instance; however all information about the process instance is still available and can, for example, be queried. Some Workflow-Management-Systems still allow operations on finished process instances, such as restarting the process instance at the beginning or even in the middle of the process instance.

No further actions can be carried out if the process instance is in the state **deleted** 204.

The state **suspended** 205 is entered as the result of entering the SUSPEND function by issuing a corresponding control command. In this state, the Workflow-Management-System stops navigation until a user via the RESUME control command requests continuation.

A process instance enters the state **terminated** 206 as the result of the TERMINATE control command, which causes the

Workflow-Management-System to stop processing the process instance.

4.3 Controlling the Creation of Process Instances

Fig. 3 illustrates how the present invention can be implemented using the Flow Definition Language (FDL) of MQSeries Workflow, a state-of-the-art Workflow-Management-System sold by the applicant. FDL is used as an example only; any other way of specifying the control of process instance creation can be used. The underlying meta model is also for illustration only; other meta models can be used instead.

The DATASTRUCTURE definition 301 identifies a data structure with the name OrderProcessInput 302 that consists of two members, a first member ISBN of data type STRING 303 and a second member CUSTOMER_ID of data type STRING 104. This data structure defines the input parameters (that is, the input container) of the business process as apparent from the interface definition 322 of the process BookOrder 321.

The PROGRAM definition 311 declares a program with the name SendEMail 312 which accepts the two fields CUSTOMER_ID and ISBN. For simplification, no further definitions, such as the actual implementation of the program, is provided.

The PROCESS definition 321 defines a process model with the name BookOrder, which accepts the previously defined data structure OrderProcessInput 322. That means, when a process instance is created via the appropriate request, the requester can (in fact should) specify concrete values for the fields in the data structure: a value for the field ISBN, and a value for the field CUSTOMER_ID. The IDENTIFIER keyword 323 is used to specify how the process instance identifier is to be constructed; in this case by combining the values of the fields CUSTOMER_ID and ISBN.

In the specific example of Fig. 3 all parameters of the input container 302 are used to create the process instance identifier; in a certain sense this is an extreme example as the IDENTIFIER specification supports to use any subset of the parameters within the input container for constructing the process instance identifier. It would even be possible to include in the IDENTIFIER specification parameters, which are not comprised by the input container, their corresponding values being provided from other sources.

Finally, the process instance identifier can be viewed as a concatenation of the concrete values of the parameters specified within the IDENTIFIER section of the process model's FDL. In a further embodiment of the current invention the name of the process (in the current example BookOrder) could automatically and implicitly be made part of the process instance identifier guaranteeing uniqueness of the process instance identifier not only within the set of process instances derived from a particular process model but within all process instances independently from the particular process model.

The keyword MANDATORY 325 is used to indicate that the requestor must supply values for the specified fields. This is necessary to make sure that the Workflow-Management-System does not generate a process instance identifier, which is meaningless, such as an empty process instance identifier; in other words, the parameters comprised within the MANDATORY specification defines the "Minimum" components whose values making up the process instance identifier.

At runtime the specifications regarding the construction of the process instance identifier are used by the WFMS in the following manner:

First it is checked whether concrete values have been specified by a requester for the parameters as specified in the MANDATORY section 325 of the process model's FDL. Without concrete values for all of the mandatory parameters no unique process instance can be created and therefore the corresponding request for creating a process instance will be rejected.

If values for the mandatory parameters have been specified with the process instance creation request a process instance identifier is constructed according to the definition specification in the IDENTIFIER section 323 (optionally comprising the process name as part of the process instance identifier). The WFMS will then check whether a process instance with that process instance identifier exists already (or has existed and was executed in the past); a new process instance for that process model will be created by the WFMS only, if that has not been the case according to that check thus establishing a "Once and only once" execution scheme.

If a process instance with this process instance identifier exists already, the action specified via the DUPLICATE keyword 304 is being carried out, in this case the invocation of the program SendEMail with the two fields as parameters. Thus based on the DUPLICATE keyword a kind of error handling is established. In the example of Fig. 3 the error handling consists in an e-mail notification on the duplicate creation request for the identical business process.

In a further embodiment of the current invention that specifications comprised by the IDENTIFIER keyword, the

DUPLICATE keyword or the MANDATORY make also be specified with the request to create a certain process instance in addition to the actual input data to be processed by the process instance. Such an approach allows for a highly dynamically manner of creating process instance identifiers and controlling the creation of process instances.

C L A I M S

1. A computerized method of providing process instance creation control within an execution environment,

   said method upon receiving an request to create a process instance,

   in a first step constructing a process instance identifier that uniquely identifies said process instance to be created based on input data provided with said request and said input data to be processed by said process instance; and

   in a second-step creating said process instance with said process instance identifier within said execution environment if no process instance exists or existed with said process instance identifier.

2. A computerized method of providing process instance creation control within an execution environment according to claim 1,

   wherein said process instance is created from a process model; and

   wherein said process model is comprising a first specification (323) defining a subset of input parameters of said process model and the values of said subset of said input parameters are to be used for constructing said process instance identifier; and

   wherein in said first-step said process instance identifier being created using said subset of said input parameters.

3. A computerized method of providing process instance creation control within an execution environment according to claim 2,

   wherein said process model is comprising a second specification (325) defining a mandatory subset of input parameters of said process model for which values must be specified with said request; and

   wherein in said second-step no process instance is created if any value of set mandatory subset of input parameters is not supplied with said request.

4. A computerized method of providing process instance creation control within an execution environment according to claims 2 or 3,

   wherein said process model is comprising a third specification (324) defining an action to be executed if another process instance with said created process instance identifier exists or existed already; and

   wherein in said second-step, if another process instance identifier with said created process instance identifier exists or existed already, executing said action.

5. A computerized method of providing process instance creation control within an execution environment according to any of the preceding claims 2 to 4,

   wherein said first specification and/or said second specification and/or said third specification is provided with said request.

6. A computerized method of providing process instance creation control within an execution environment according to any of the preceding claims,

   wherein said execution environment is a WFMS; and

   wherein said method is executed by the WFMS itself.

7. A computerized method of providing process instance creation control within an execution environment according to claim 6,

   wherein said request to create a process instance is carried out by the WFMS itself.

8. A system comprising means adapted for carrying out the steps of the method according to anyone of the preceding claims 1 to 7.

9. A data processing program for execution in a data processing system comprising software code portions for performing a method according to anyone of the preceding claims 1 to 7 when said program is run on said computer.

10. A computer program product stored on a computer usable medium, comprising computer readable program means for causing a computer to perform a method according to anyone of the preceding claims 1 to 7 when said program is run on said computer.

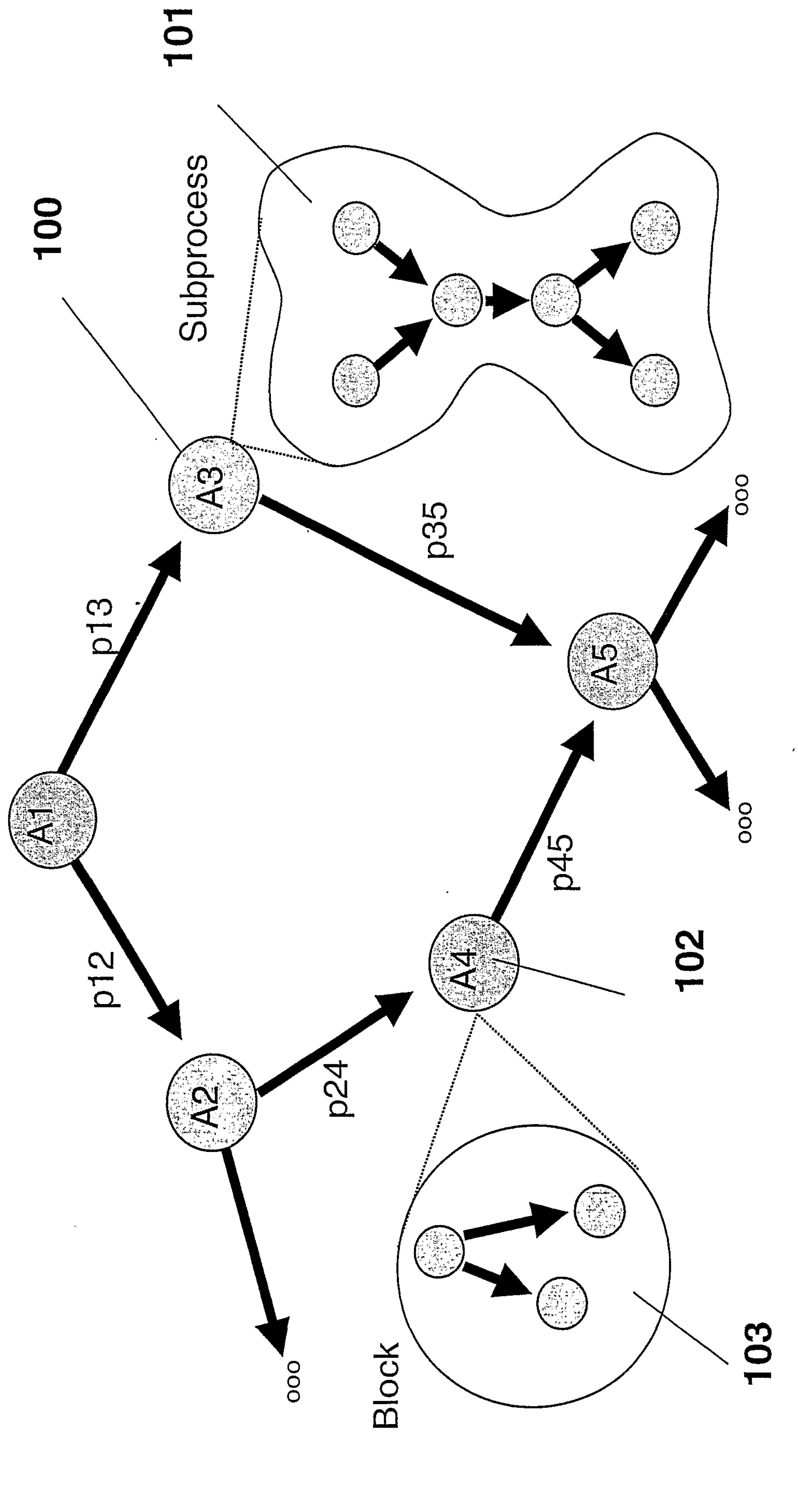

100
101
Subprocess
A1
A2
A3
A4
A5
p12
p13
p24
p35
p45
ooo
ooo
ooo
102
Block
103

FIG. 1

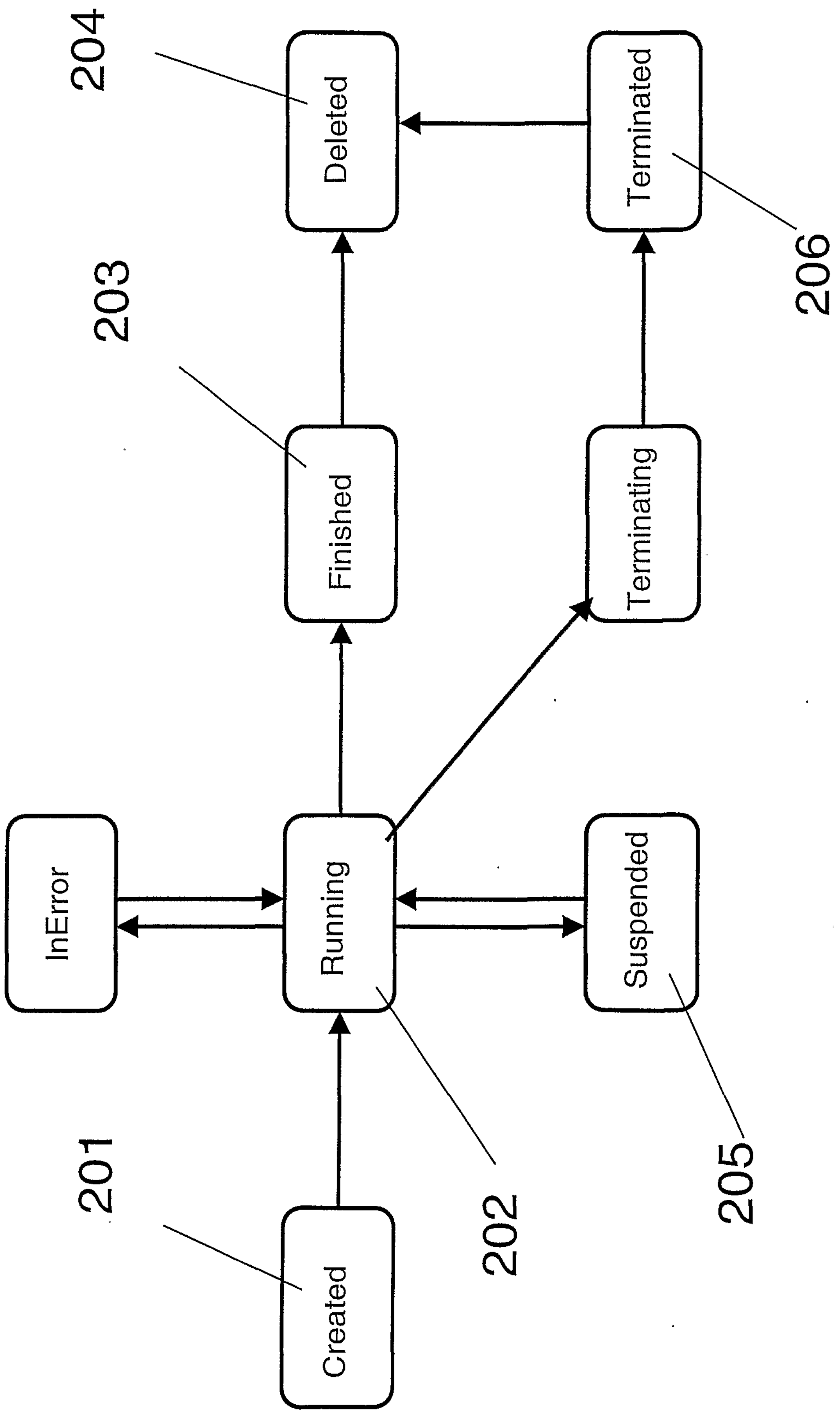
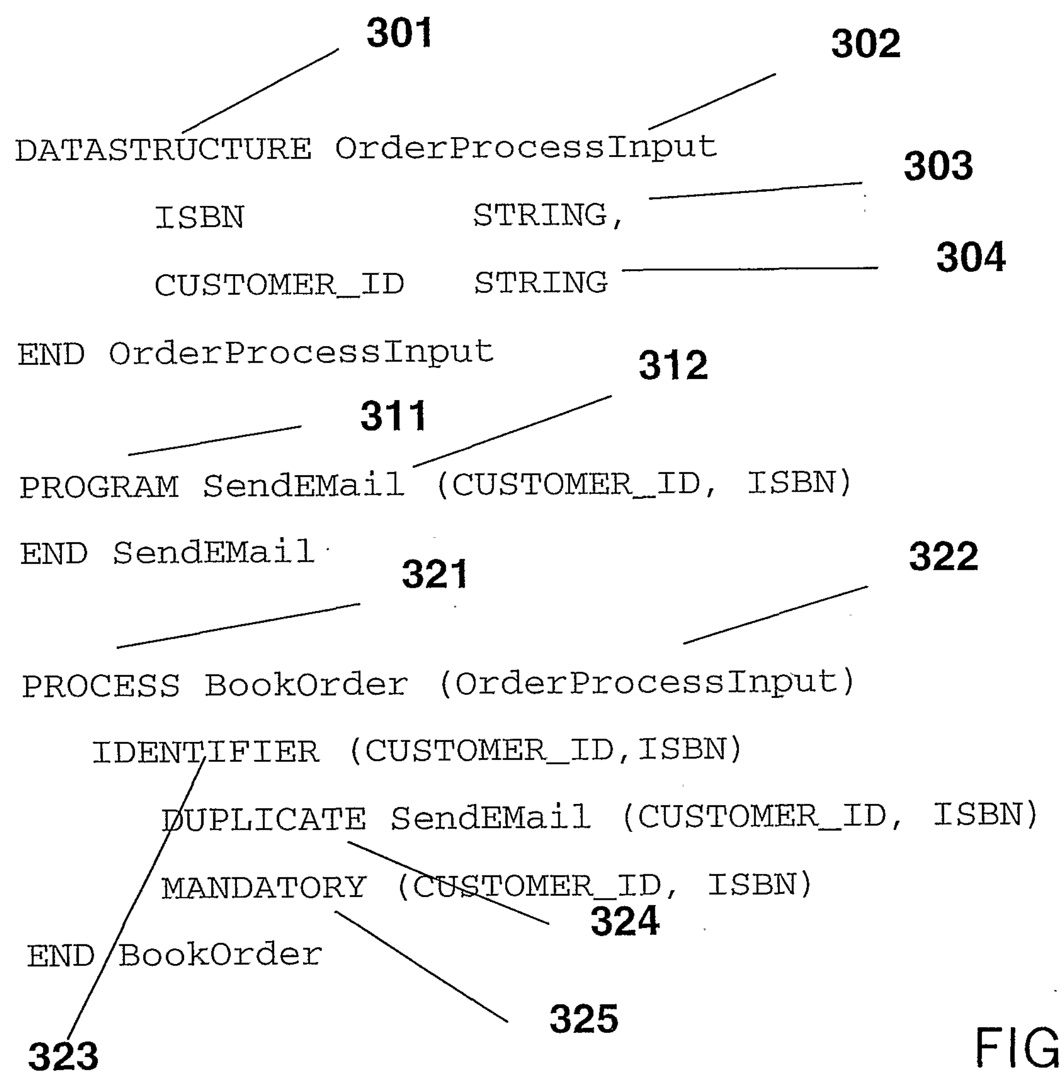

Created
201
Running
202
InError
Suspended
205
Finished
203
Terminating
Deleted
204
Terminated
206

FIG. 2



```
DATASTRUCTURE OrderProcessInput
        ISBN            STRING,
        CUSTOMER_ID     STRING
END OrderProcessInput

PROGRAM SendEMail (CUSTOMER_ID, ISBN)
END SendEMail

PROCESS BookOrder (OrderProcessInput)
    IDENTIFIER (CUSTOMER_ID,ISBN)
        DUPLICATE SendEMail (CUSTOMER_ID, ISBN)
        MANDATORY (CUSTOMER_ID, ISBN)
END BookOrder
```

FIG. 3