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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 June 2005 (16.06.2005)

PCT

(10) International Publication Number
WO 2005/055103 A1

(51) International Patent Classification⁷: G06F 17/60

(21) International Application Number: PCT/IE2004/000166

(22) International Filing Date: 3 December 2004 (03.12.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2003/0907 3 December 2003 (03.12.2003) IE

(71) Applicant *(for all designated States except US)*: **CORECT LIMITED** [IE/IE]; 3/4 Merrion Place, Dublin 2 (IE).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **BOURKE, Sarah** [IE/IE]; 1 Townhouse, Terenure Road East, Dublin 6 (IE). **KIERNAN, Paul** [IE/IE]; 3 Gowran Hall, Ballygihen Avenue, Sandycove, County Dublin (IE).

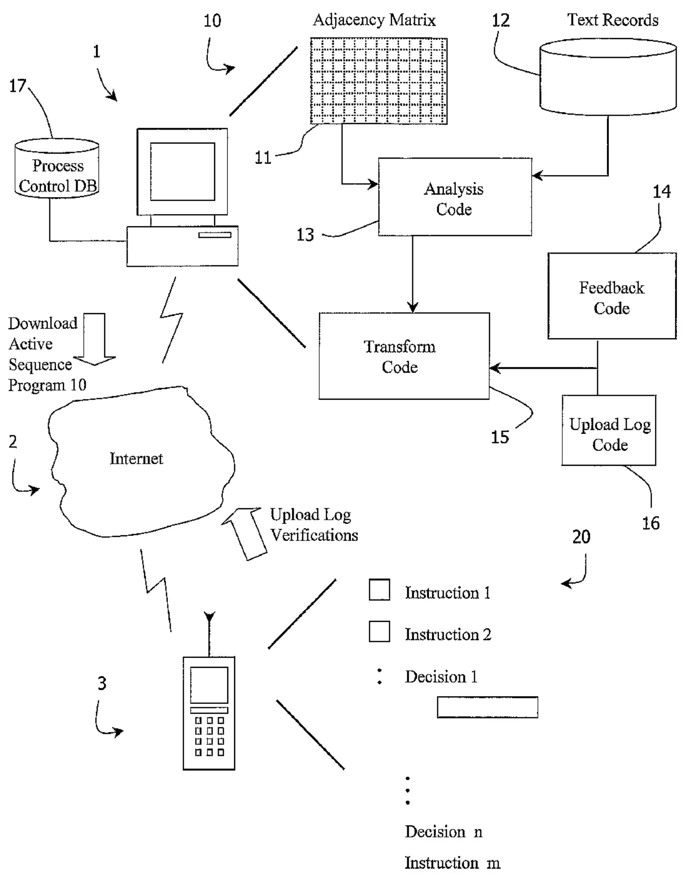
(74) Agents: **O'BRIEN, John, A.** et al.; c/o John A. O'Brien & Associates, Third Floor, Duncairn House, 14 Carysfort Avenue, Blackrock, County Dublin (IE).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: CONTROL OF PROCESSES



(57) Abstract: A server (1) automatically interprets a flow diagram process description to generate an active sequence program (10). The program (10) has analysis (13), feedback (14), transform (15), and upload (16) code which are common in core functionality across multiple programs (10). It is specific to a particular process by virtue of an adjacency matrix (11) of step links and text records (12). The program (10) is downloaded to a mobile device (3). The device (3) executes the program (10) to generate a current step instruction, to receive user feedback and automatically generate a next step. A realtime time stamp is applied to each user feedback and it is automatically uploaded to the server (1).

FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

"Control of Processes"

INTRODUCTION

Field of the Invention

The invention relates to control of processes carried out by humans. Such processes include, for example, procedures in spacecraft or aircraft, or alternatively more mundane but important procedures for cleaning/hygiene in restaurants or hotels.

Prior Art Discussion

Within industry today, engineers are required to perform operations which require complex decisions to be undertaken. For example, for maintenance of equipment engineers must try to analyse and determine the fault and then perform the correct repair process. Usually, the engineer performing the operation is not the expert in a particular field and requires remote support and assistance tools in performing the required tasks.

It is common to use task graphs which are a pictorial representation of the instructions and decisions that a person must follow. The engineer would manually "walk through" the graph following the instructional steps. The graphs contain decision steps in which a question is asked and depending on the response, a different pathway in the task graph is followed. The engineer would "walk through" the task graph until a final instruction is reached. Fig. A shows a simple task graph that a person would use to perform a process task. The different shapes within the task graph define semantic meanings. For example, a rectangle represents an instruction which must be performed and a diamond shape represents a decision step which can have multiple answers. The different outcomes in a decision instruction are defined by the answers on the lines coming out of the decision step and these define the appropriate pathway to follow.

The use of task graphs for performing complex processes has several disadvantages for an end user. A complex process can have a very large task chart associated with it which if printed can run to multiple pages. Also manual cross referencing of appropriate instruction tasks on a printed task graph can be very cumbersome. Another problem is that a user has to view the complete process with all possible decisions. Also, viewing an electronic version of a task chart on portable computer equipment is awkward and slow.

The invention addresses this problem.

SUMMARY OF THE INVENTION

According to the invention, there is provided a method for process control, the method comprising:

> an interpretation function receiving a description of a process and interpreting the process description to generate an active sequence program, said active sequence program comprising stored process flow data and analysis program code; and
>
> a run-time processor executing the active sequence program code to analyse said data and generate a sequence of instructions to advance the process to completion in which a current instruction is automatically indicated.

In one embodiment, the process description is a flow chart.

In another embodiment, the process description is converted to mark-up language format, and the mark-up language is processed to provide the active sequence program.

In a further embodiment, the interpretation function processes the mark-up language to provide a document object model (DOM) representation of the process description.

In one embodiment, the interpretation function processes each node of the document object model to determine if it relates to an instruction step, to a decision step or to a link between steps.

In another embodiment, the interpretation function generates process flow data in the form of an adjacency matrix defining links between the steps.

In a further embodiment, the interpretation function extracts text from each node representing an instruction step or a decision step and saves the text strings in an addressable text record set.

In one embodiment, the interpretation function saves the adjacency matrix, the text record set and pre-configured blocks of analysis, feedback, and transform program code to provide the active sequence program.

In another embodiment, the text records are an array of HTML strings.

In a further embodiment, the strings are labelled according to a number for the step.

In one embodiment, the interpretation function generates the active sequence program in a format suitable for download to the processor via a wide area network, and the method comprises the further step of downloading it to a remote processor via the wide area network.

In another embodiment, the interpretation function operates in real time in response to a request from a remote client device and it generates the active sequence program to be suitable to execute on the processor of the requesting client device.

In a further embodiment, the active sequence program is wrapped in a HTML page having a body element with a reference to an initialisation script for initialisation of the program on the processor.

In one embodiment, the processor executes the active sequence program to:

display text for a current step;

display a feedback prompt for a user to either indicate performance of the step if it is an instruction step, or an answer if it is a decision step; and

receive feedback and automatically determine a next step and display text for the next step.

In another embodiment, the processor automatically time stamps each feedback input from a user.

In a further embodiment, the processor automatically uploads the user feedback and time stamp to a remote server.

In one embodiment, the server is the same server as that which executes the interpretation function.

In another embodiment, the server maintains a monitoring log of performance of process steps with real-time time stamps.

In a further embodiment, the processor can be switched to a free mode in which steps can be skipped for training purposes.

In one embodiment, a transform code block of the active sequence program receives a batch of analysed data from an analysis code block, and determines a current step according to inputs from a feedback code block.

In another embodiment, the transform code uses style sheets to change display characteristics.

In a further embodiment, the processor dynamically determines according to the process flow data the maximum number of instructions which can be displayed.

In one embodiment, if the processor receives an interrupt before it records performance of a final step of the process, it automatically saves a process status data set and uploads said data set to a server.

In another embodiment, the process status data set is passed to a method within a hidden applet, and the hidden applet opens a URL connection and encodes and transmits the status data to a servlet on the server.

In a further embodiment, the processor automatically checks with a server for existence of a relevant process status set when it receives a request to activate a process.

In another aspect of the invention, there is provided a server whenever programmed to generate an active sequence program in a method as described above.

In a further aspect of the invention, there is provided a computer program product comprising software code for performing the steps of a method as described above when executing on a digital processor.

DETAILED DESCRIPTION OF THE INVENTION

Brief Description of the Drawings

The invention will be more clearly understood from the following description of some embodiments thereof, given by way of example only with reference to the accompanying drawings in which:-

Fig. 1 is a diagram illustrating a control system of the invention;

Fig. 2 is a flow diagram illustrating both initialisation and run-time operations of the system; and

Fig. 3 is a screen display for an active sequence generated by the system.

Description of the Embodiments

Referring to Fig. 1 a server 1 generates an active sequence program 10 from a received flow diagram, and downloads it via the internet 2 to a mobile device 3 for execution. As steps are performed at the user end, the device 3 automatically uploads a log verification to the server 1. The server 1 therefore performs real time process monitoring.

The active sequence program 10 comprises an adjacency matrix 11, a text record set 12, analysis code 13, feedback code 14, transform code 15, and update log code 16. Briefly, the program 10 is generated by the server 1 and is downloaded to the mobile device 3 of a person who is to carry out a process. The program 10 executes on the mobile device 3 and generates an active sequence 20 of instruction and decision steps. These are described in more detail below. The user may use any suitable computing device. As steps are performed the user indicates this by feedback on the device 3, which automatically transmits an upload to the server 1 on a step-by-step basis for saving in a process control database 17.

Referring to Fig. 2, in an initialisation mode the server 1 executes a process involving editing of a flow diagram in any suitable flow chart application such as Microsoft Visio™ using an editor 25. An interpretation function 27 executes to generate the program 10. Alternatively, the initialisation mode may involve editing a word processor document and the interpretation function 27 processing it to generate the program 10. The program 10 is then stored in a store 28 for download in response to an uploaded request from a mobile device.

The editor 25 defines the process by connecting graphical elements with connectors which route through basic shapes. The connectors are used to define the flow through a process and the pathways that are available within a process. The connectors can either be pathways that must be followed or they can be conditional based on information supplied with the connector such as Yes/No, True/False or answers to multiple choice questions. Conditional connectors are connected to a decision step within a process flow diagram. On completion, there is a graphical flow chart representation of a process as shown in Fig. A.

Referring again to Fig. 1, the program 10 comprises four main blocks of code, namely the analysis code 13, the feedback code 14, the transform code 15, and the upload log code 16. These are essentially common across multiple programs 10. However the interpretation function 27 generates from the received flow diagram the adjacency matrix 11, defining links between the instruction and decision steps of the received flow chart. It also extracts text from the instructions and decisions of the received flow chart to provide the text records 12. Thus, each active sequence program 10 comprises common code blocks 13, 14, 15, and 16, and a specific adjacency matrix 11 and text record set 12.

Referring to Fig. 3 the active sequence corresponding to the edited flow chart of Fig. A is represented as a sequence of instructions. Each instruction has a check box for user feedback. Each decision step of the flow chart is represented by a question and a drop-down list of options. The underlying code 13, 14, and 15 generates the drop-down list options, and acts upon selection of an option to activate selected ones of the remaining steps. Activation of a subsequent step is achieved by simply activating its check box.

Thus, the interpretation function 27 converts passive flow diagram elements into active check boxes, text, drop-down lists and underlying code to activate subsequent steps. The complexity of the flow branching is transparent to the user, who is only required to respond to a current step. The user in run-time contributes dynamically to the selection of current step, although he or she does not appreciate this.

Referring again to Fig. 2, in run-time, on the mobile device 3 the sequence is outputted in step 35, and feedback is received in step 36. This feedback (checking of boxes and choice of drop-down list options) both allows the system to dynamically modify the down-stream steps, and to record and upload process status. This involves time-stamping the steps as they are performed.

Step 37 involves verifying the feedback and the code 16 uploading it to the server 1. Step 18 involves modifying the downstream steps. As indicated by steps 39 and 40, the program 10 in runtime verifies, records, and modifies repeatedly until the process is complete. When complete, the program 10 terminates, and the server 1 has recorded a log of all process steps with real-time time stamps.

In more detail, the function 27 is activated by being passed a pointer to an XML document for a flow chart. It then parses the XML document to generate a document object model (DOM) representation of the flow diagram. The DOM has a hierarchy of nodes, each of which is tagged. The function 27 then traverses the nodes and, according to the tags of the nodes allocates each node to an instruction step, a decision step, or a link. It uses these allocations to generate the adjacency matrix 11, and for the nodes corresponding to decision or instruction steps it saves the text to the text record set 12. The function 27 couples the matrix 11 and the text records 12 with the program code blocks 13, 14, 15, and 16 to provide the complete active sequence program 10, in executable Java Script format.

The two dimensional adjacency matrix 11 represents connectivity information between each instruction within the task chart. The adjacency matrix (A) is defined as follows: Let G be a graph with "n" vertices that are assumed to be ordered from v_1 to v_n. The n x n matrix A, in which

$a_{ij}= 1$ if there exists a path from v_i to v_j

$a_{ij} = 0$ otherwise

The records 12 are an array of strings containing the HTML which represents each instruction within the procedure. Each instruction is represented by a HTML <TABLE> where the **name** attribute of a table is a unique value (i) which represents that it is the i^{th} instruction in the XML document. The HTML represents decision steps graphically with a drop-down list component. The list contains the text associated with each line coming out of the decision step. Each piece of HTML also contains a check box element which is used to verify the step.

The program 10 also comprises String Variables initialised to general metadata information about the instruction chart such as process title, process purpose and author. The analysis code 13 interprets the adjacency matrix 11 and the transform code 15 displays the appropriate instructions in a sequential manner within a HTML document. The transform code 15 receives inputs from the feedback code 14 to maintain the 'current' step of the process. Essentially, the analysis code 13 generates process data in a batch from analysis of the matrix 11 and text records 12, and passes the batch to the transform code 15. The transform code 13 includes functions to store the pathway within the task graph chosen by the user during execution. The code 16 logs execution information back to the web server and functions to interpret and restore the state of an executing procedure to the state a procedure was previously left at.

The generated program 10 is wrapped within a HTML page which has an empty <BODY> element. This BODY element has a reference to an initialisation JavaScript method via an onLoad method, for interpretation and display of the task when the page is completely loaded on the client's browser.

The created HTML page is streamed back from the web server to the web browser (step 28) via the transformation Servlet.

During the transformation process the Servlet will identify if there are logical inconsistencies within the procedure. For example, if a decision step doesn't have unique answers on line out connectors or if instructions exist that cannot be reached via a unique path. The user is informed of this during the transformation process, and is a useful tool for validation of the correctness within the definition of the underlying task chart.

Active sequence display and execution (Steps 35-40)

A user who wishes to execute a task chart launches their web browser and accesses the server 1. The user is presented with a login screen where they must enter a valid username/password combination to access the system. The username is then used for logging purposes during task execution.

The server 1 transmits in real time the generated JavaScript program 10 to end user devices where it is executed (steps 35-40) by a standard web browser or other standard JavaScript interpreter application.

The user device displays the task instructions as a series of sequential instructions within a standard web browser window (Fig. 3). The device 3 displays only the path within the task chart that the user is currently within, based on earlier decisions that they have made while executing the task. It displays as many instruction tasks in advance as are logically possible up to a future decision step. The user is allowed to stop and restart a procedure in the same state during a subsequent session. The device 3 in real time logs and transmits back to the server 1 all decisions made and the date/time of all instructions executed by the user.

When the page is interpreted by the device 3, metadata information on the process is automatically interpreted and displayed by JavaScript functions of the transform code 15. The look of metadata information is described by style sheets which are referenced by the HTML page.

After displaying the task metadata, the server 1 is queried to see if state information for this procedure exists. If state information for the procedure exists this information is loaded and is used to restart the procedures and outline in the State storage section. Otherwise, the adjacency matrix 11 is analysed by the generated JavaScript routines 13 to determine the initial root instruction of the procedure. This step is marked as the current instruction within the executing procedure. The adjacency matrix 11 is then processed to determine a unique pathway up to a decision step. At a decision step no further steps can be added to the procedure without user intervention.

The list of instruction steps that can be displayed is then transformed into a sequence of HTML code fragments, each of which represents an instruction. The HTML code is then dynamically inserted into the HTML DOM so that they are displayed in sequence on the HTML page to the user. The look of an instruction is defined by reference to an external style sheet file.

The current instruction is highlighted with a different background colour. A current instruction is the only instruction that the user can verify. To verify an instruction the user can either click within the check box that is contained in each instruction or use a predefined short cut to verify the current instruction.

When an instruction is verified (step 36), the current step is advanced to the next step as determined by the unique path identified through the analysis of the adjacency matrix 11. This new current step is then highlighted.

When a decision instruction is reached within an executing procedure and is the current step, no further instructions are displayed in the web browser. The user must choose the appropriate pathway to follow in the task chart for more steps to be displayed. This is done by the user choosing an entry within a drop down list which is displayed within an instruction representing a decision step. Each entry in the drop down list is the value of text associated with a line connector coming out of the decision instruction within the task chart. Once the user selects an entry and verifies the decision step, the JavaScript code 13 can determine which pathway to follow within the adjacency matrix 11. This pathway is then analysed to determine the maximum number of instructions that can be displayed until the next decision step is reached. These next block of steps are then displayed underneath the steps already displayed.

With this approach, the instructions are displayed in the sequential manner for execution. At no stage are instructions removed from the list. If the task chart loops back to a previous step, the same steps are redisplayed further down the checklist display.

At no stage during the execution process does the device 3 require communication with the server 1 to retrieve further information about the process. Full information about the process is contained with the HTML page.

The user can continue executing the process until an instruction is displayed which has no line connectors coming out of it within the task chart description. This represents a final instruction within the task.

Forced and Free Sequence Execution

The program 10 operates by default in a forced sequence execution. A single instruction is defined as the current instruction. This is the only instruction that the user can execute. However, there is also a "free" mode in which steps can be skipped. This mode is mainly for training purposes.

The mode is selected using a toggle button displayed within a separate page within a frameset. When executing a procedure in 'forced' sequence mode, all previous instructions in the sequential display must be verified before the current instruction can be verified. When executing a procedure in 'free' sequence mode, a user can verify a current instruction without all previous instructions being verified. The user can change the current instruction in 'free' sequence mode by clicking on any of the procedure steps. In 'forced' sequence mode the user cannot change the current step interactively. Within both modes if the user verifies a decision instruction, the next series of appropriate instructions are displayed.

Logging

The upload log code 16 provides for the logging of all instructions verified by the user in real time by the server 1. Upon initialisation of the HTML page by the web client a reference to a custom applet (the code 16), developed for this process, is inserted into the HTML page. This applet is hidden from the user. The applet contains methods which:

1. Can be called from JavaScript functions within the page
2. Can call JavaScript functions within the page.

The custom applet has two main categories of features.

1. Methods for logging of user instruction verification
2. Methods for storage/retrieving of procedure state information.

Each time the user verifies a step, a JavaScript method within the HTML page is called. This JavaScript method gathers information such as the username, date and time of verification, unique instruction step identifier and passes this information via a call to a method within the hidden applet. The applet takes this information, opens a connection to the web server and calls a URL encoding this information as parameters to the URL. The called URL invokes a Servlet which decodes the passed information and stores the log information within a database on the web server. This mechanism provides for real time transmission of task execution information for storage on a centralised server.

State storage and recovery

The program 10 provides the ability to stop and restart the execution of a process by a user at a later stage. If a user exits a process before reaching a final instruction, JavaScript methods are called on exiting which creates the following information:

1. An array of all identifiers of instructions that have been displayed to the user.
2. Whether the instruction was verified or not.
3. Identifier of current instruction.
4. Username.
5. Procedure metadata

This information is then passed to a method within a hidden applet. The applet opens a URL connection where the information is encoded and passed to a Servlet executing on the web server. The Servlet decodes the information and stores this information within an XML file on the web server. The file is named based on the procedure metadata and user name.

When a process is initialised for display, the JavaScript methods call the hidden applet to determine if a state file for this procedure and user exist on the server. The applet connects via a URL to a Servlet which determines if a state file exists. If a state file exists it is interpreted and returns the state information to the applet. The applet in turn sends the information back to the calling JavaScript method. JavaScript methods are then used to extract the information and recreate the instructions within the HTML document alone with the verification state of each instruction. Finally the appropriate instruction within the procedure is made the current instruction. The process is then rendered from this point.

It will be appreciated that the invention provides for controlled performance of processes at remote sites in a simple and convenient manner, together with real time monitoring on a step-by-step basis.

The invention is not limited to the embodiments described but may be varied in construction and detail.

Claims

1. A method for process control, the method comprising:

   an interpretation function (27) receiving a description of a process and interpreting the process description to generate an active sequence program, said active sequence program (10) comprising stored process flow data (11, 12) and analysis program code (13-16); and

   a run-time processor (3) executing the active sequence program code (13-16) to analyse said data and generate a sequence of instructions to advance the process to completion in which a current instruction is automatically indicated.

2. A method as claimed in claim 1, wherein the process description is a flow chart.

3. A method as claimed in claims 1 or 2, wherein the process description is converted to mark-up language format, and the mark-up language is processed to provide the active sequence program.

4. A method as claimed in claim 3, wherein the interpretation function (27) processes the mark-up language to provide a document object model (DOM) representation of the process description.

5. A method as claimed in claim 4, wherein the interpretation function (27) processes each node of the document object model to determine if it relates to an instruction step, to a decision step or to a link between steps.

6. A method as claimed in claim 5, wherein the interpretation function (27) generates process flow data in the form of an adjacency matrix (11) defining links between the steps.

7. A method as claimed in claim 6, wherein the interpretation function (27) extracts text from each node representing an instruction step or a decision step and saves the text strings in an addressable text record set (12).

8. A method as claimed in claim 7, wherein the interpretation function (27) saves the adjacency matrix (11), the text record set (12) and pre-configured blocks of analysis (13), feedback (14), and transform (15) program code to provide the active sequence program.

9. A method as claimed in claims 7 or 8, wherein the text records (12) are an array of HTML strings.

10. A method as claimed in claim 9, wherein the strings are labelled according to a number for the step.

11. A method as claimed in any preceding claim, wherein the interpretation function (27) generates the active sequence program (10) in a format suitable for download to the processor via a wide area network (2), and the method comprises the further step of downloading it to a remote processor (3) via the wide area network.

12. A method as claimed in claim 11, wherein the interpretation function (27) operates in real time in response to a request from a remote client device (3) and it generates the active sequence program (10) to be suitable to execute on the processor of the requesting client device (3).

13. A method as claimed in any preceding claim, wherein the active sequence program (10) is wrapped in a HTML page having a body element with a reference to an initialisation script for initialisation of the program on the processor.

14. A method as claimed in any preceding claim, wherein the processor executes the active sequence program (10) to:

display (35) text for a current step;

display (35) a feedback prompt for a user to either indicate performance of the step if it is an instruction step, or an answer if it is a decision step; and

receive feedback (36) and automatically determine a next step and display text for the next step.

15. A method as claimed in claim 14, wherein the processor automatically time stamps (37) each feedback input from a user.

16. A method as claimed in claim 15, wherein the processor automatically uploads (37) the user feedback and time stamp to a remote server (1).

17. A method as claimed in claim 16, wherein the server is the same server (1) as that which executes the interpretation function (27).

18. A method as claimed in claim 17, wherein the server maintains a monitoring log (17) of performance of process steps with real-time time stamps.

19. A method as claimed in any preceding claim, wherein the processor can be switched to a free mode in which steps can be skipped for training purposes.

20. A method as claimed in any of claims 14 to 19, wherein a transform (15) code block of the active sequence program (10) receives a batch of analysed data from an analysis code block (13), and determines a current step according to inputs from a feedback code block (14).

21. A method as claimed in claim 20, wherein the transform code (15) uses style sheets to change display characteristics.

22. A method as claimed in any preceding claim, wherein the processor dynamically determines according to the process flow data the maximum number of instructions which can be displayed.

23. A method as claimed in any preceding claim, wherein if the processor receives an interrupt before it records performance of a final step of the process, it automatically saves process status data set and uploads said data set to a server.

24. A method as claimed in claim 23, wherein the process status data set is passed to a method within a hidden applet, and the hidden applet opens a URL connection and encodes and transmits the status data to a servlet on the server.

25. A method as claimed in claims 23 or 24, wherein the processor automatically checks with a server for existence of a relevant process status set when it receives a request to activate a process.

26. A server whenever programmed to generate an active sequence program in a method as claimed in any preceding claim.

27. A computer program product comprising software code for performing the steps of a method of any preceding claim when executing on a digital processor.

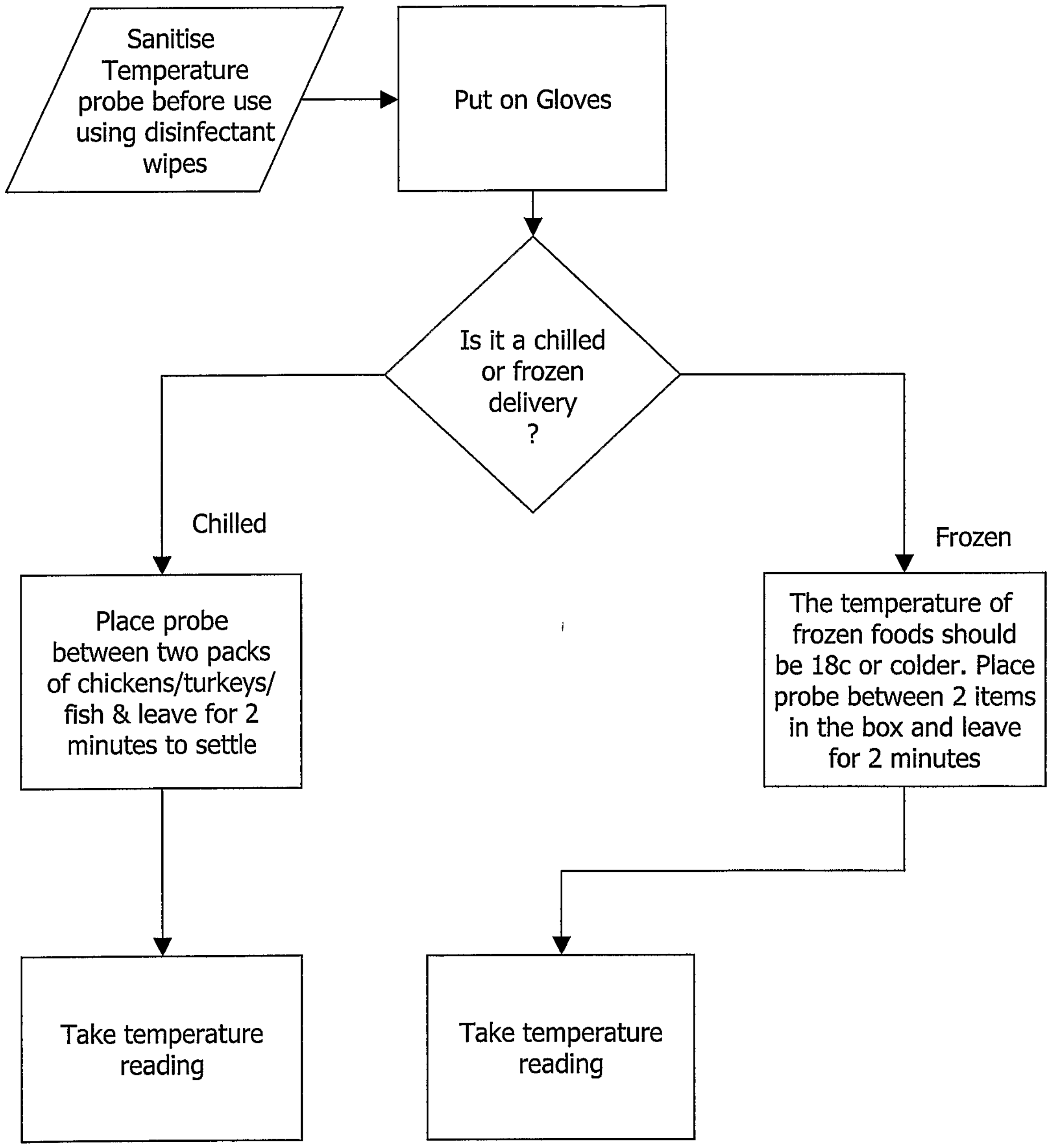

Sanitise Temperature probe before use using disinfectant wipes
Put on Gloves
Is it a chilled or frozen delivery ?
Chilled
Frozen
Place probe between two packs of chickens/turkeys/ fish & leave for 2 minutes to settle
The temperature of frozen foods should be 18c or colder. Place probe between 2 items in the box and leave for 2 minutes
Take temperature reading
Take temperature reading

Fig. A Prior Art

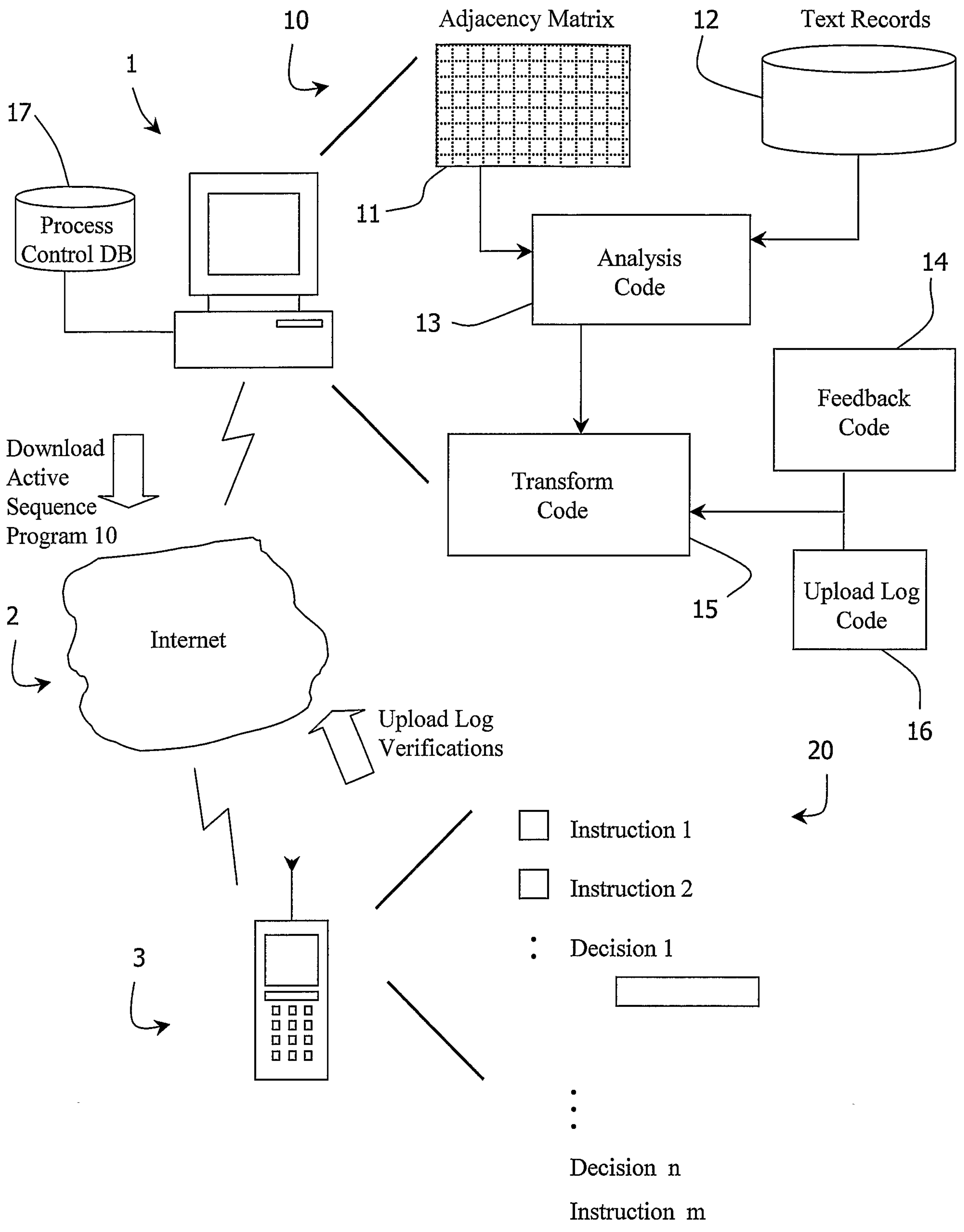

10
Adjacency Matrix
12
Text Records
1
17
Process Control DB
11
Analysis Code
13
14
Feedback Code
Download Active Sequence Program 10
Transform Code
Internet
2
15
Upload Log Code
16
Upload Log Verifications
20
Instruction 1
Instruction 2
Decision 1
3
Decision n
Instruction m

Fig. 1

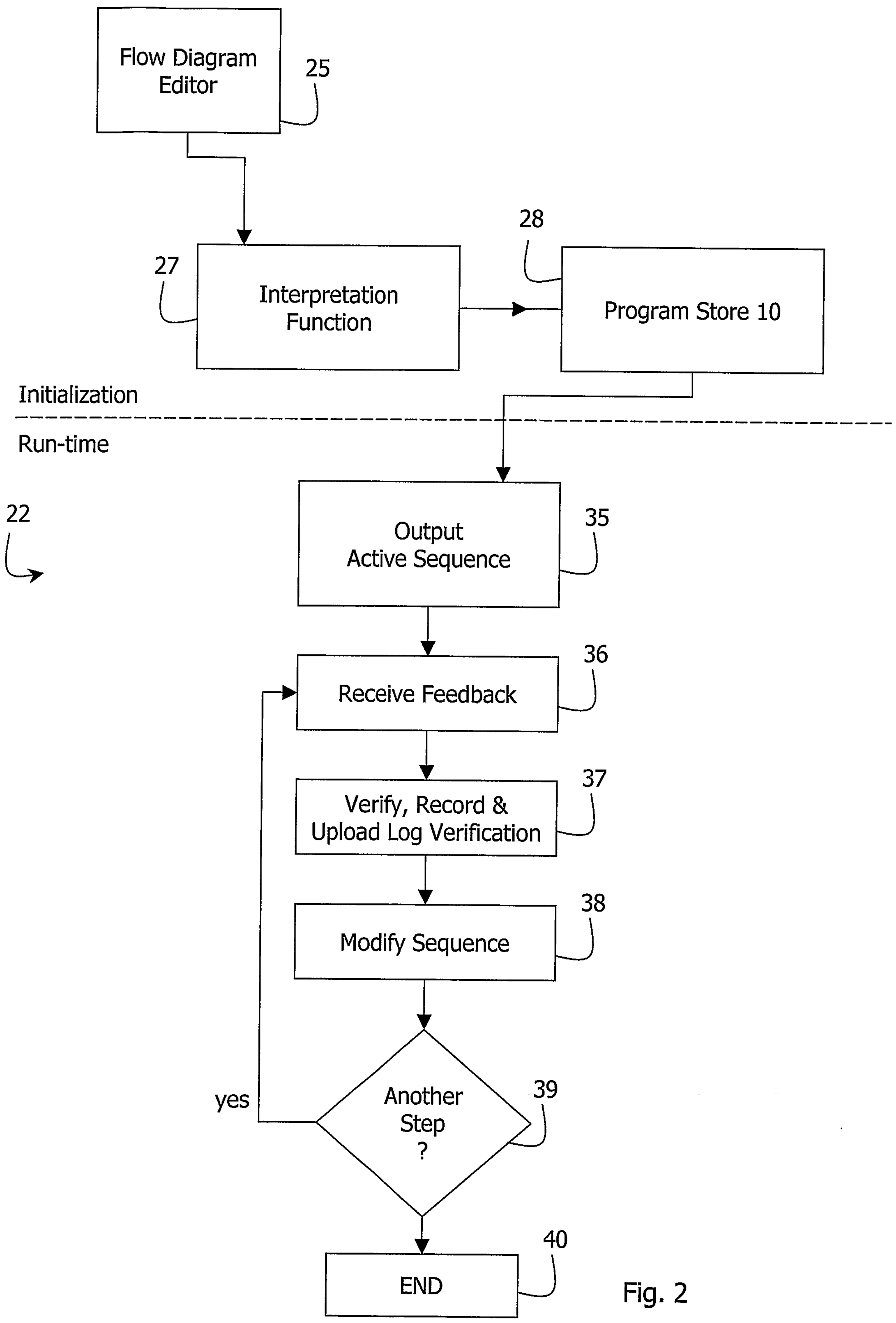
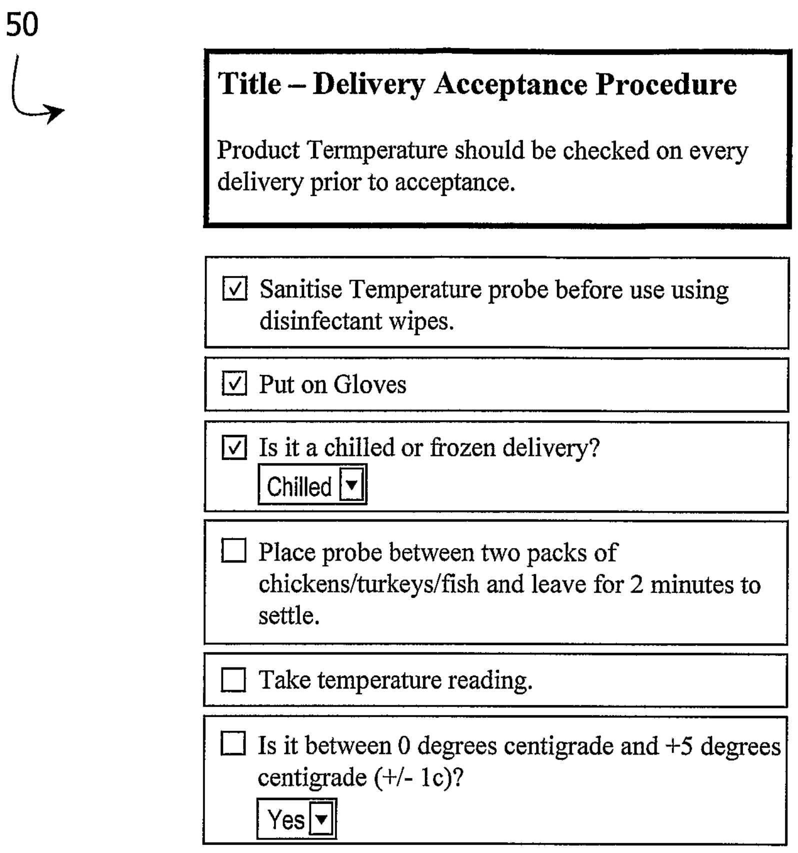

Flow Diagram Editor
25
27
Interpretation Function
28
Program Store 10
Initialization
Run-time
22
Output Active Sequence
35
Receive Feedback
36
Verify, Record & Upload Log Verification
37
Modify Sequence
38
yes
Another Step ?
39
END
40

Fig. 2



Title – Delivery Acceptance Procedure

Product Termperature should be checked on every delivery prior to acceptance.

- [✓] Sanitise Temperature probe before use using disinfectant wipes.
- [✓] Put on Gloves
- [✓] Is it a chilled or frozen delivery? Chilled ▾
- ☐ Place probe between two packs of chickens/turkeys/fish and leave for 2 minutes to settle.
- ☐ Take temperature reading.
- ☐ Is it between 0 degrees centigrade and +5 degrees centigrade (+/- 1c)? Yes ▾

Fig. 3

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IE2004/000166

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/60

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

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

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, IBM-TDB, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 831 580 A (YAMADA ET AL) 16 May 1989 (1989-05-16) * Abstract *claims 1-4 column 2, line 3 - line 59 -----	1-27
X	US 2002/165876 A1 (ENGHAUSER PETER ET AL) 7 November 2002 (2002-11-07) * Abstract * paragraph ‘0001! - paragraph ‘0013! claims 1-7 -----	1-27
X	WO 03/093988 A1 (CEDAR POINT COMMUNICATIONS INC; DEVINE, GEOFFREY; QUIGLEY, PATRICK; WE) 13 November 2003 (2003-11-13) abstract claims 1-28 ----- -/--	1-27

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search 25 January 2005	Date of mailing of the international search report 02/02/2005
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Authorized officer Daman, M

1

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IE2004/000166

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 386 508 A (ITONORI ET AL) 31 January 1995 (1995-01-31) * Abstract * column 1, line 45 - column 2, line 29	1-27
A	US 5 640 501 A (TURPIN ET AL) 17 June 1997 (1997-06-17) * Abstract * figures 1-34 column 1, line 66 - column 4, line 15	1,26,27

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/IE2004/000166

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4831580 A	16-05-1989	JP 62282301 A	08-12-1987
US 2002165876 A1	07-11-2002	DE 10110208 A1	19-09-2002
		DE 50200344 D1	19-05-2004
		EP 1237118 A1	04-09-2002
WO 03093988 A1	13-11-2003	WO 03094453 A1	13-11-2003
		WO 03094459 A1	13-11-2003
		US 2004008724 A1	15-01-2004
		US 2003217190 A1	20-11-2003
		US 2004008718 A1	15-01-2004
		WO 2004051497 A1	17-06-2004
US 5386508 A	31-01-1995	JP 2035299 C	28-03-1996
		JP 4104324 A	06-04-1992
		JP 7072861 B	02-08-1995
US 5640501 A	17-06-1997	AT 158427 T	15-10-1997
		CA 2054026 A1	01-05-1992
		DE 69127672 D1	23-10-1997
		DE 69127672 T2	07-05-1998
		EP 0483664 A2	06-05-1992
		US 5742836 A	21-04-1998
		US 5745712 A	28-04-1998
		US 5608898 A	04-03-1997