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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 September 2004 (16.09.2004)

PCT

(10) International Publication Number
WO 2004/079510 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number:
PCT/US2004/005260

(22) International Filing Date: 23 February 2004 (23.02.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/451,313	28 February 2003 (28.02.2003)	US
10/780,299	17 February 2004 (17.02.2004)	US

(71) Applicant *(for all designated States except US)*: **BEA SYSTEMS INC.** [US/US]; 2315 North First Street, San Jose, California 95131 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **CALAHAN, Patrick** [US/US]; 174 Beaver Street, San Francisco, California 94114 (US).

(74) Agents: **MEYER, Sheldon, R.** et al.; FLIESLER MEYER LLP, Four Embarcadero Center, Fourth Floor, San Francisco, California 94111 (US).

(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.

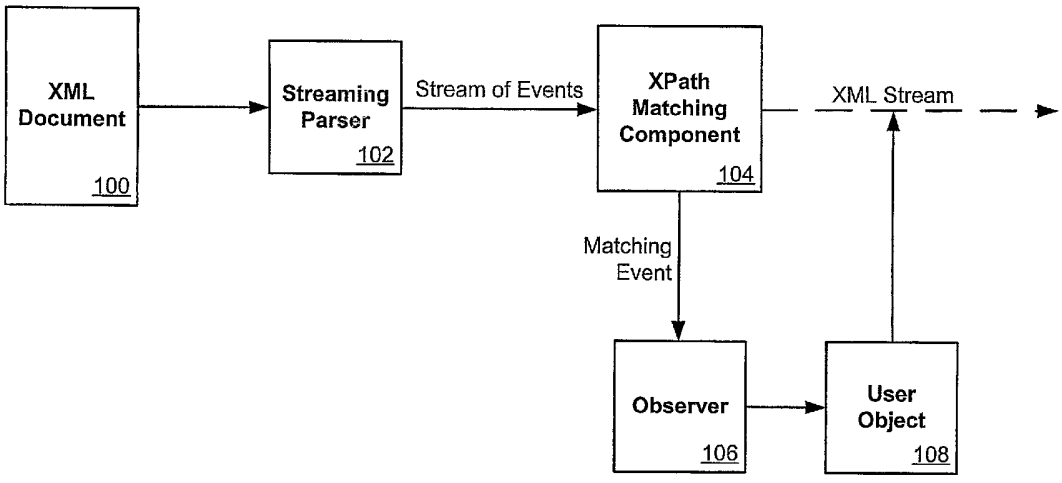
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations 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, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD,*

[Continued on next page]

(54) Title: SYSTEMS AND METHODS FOR STREAMING XPATH QUERY



(57) Abstract: ABSTRACT An improved XML query system represents an XML document as a stream of discrete 'events,' with each event representing a portion of the document as the document is being parsed. Expression-based event matching such as XPath can be performed against the event stream using a stack to keep only the relevant contexts in memory. Observers can be used to listen for matching events. Matching events can then be routed for processing by appropriate objects or components and returned to the event stream if necessary. This description is not intended to be a complete description of, or limit the scope of, the invention. Other features, aspects, and objects of the invention can be obtained from a review of the specification, the figures, and the claims.

WO 2004/079510 A2

SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

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.

SYSTEMS AND METHODS FOR STREAMING XPATH QUERY

COPYRIGHT NOTICE

CLAIM OF PRIORITY

This application claims priority from the following applications, which are hereby incorporated by reference in their entirety:

U.S. Provisional Application No. 60/451,313, entitled SYSTEMS AND METHODS FOR STREAMING XPATH QUERY, by Patrick Calahan, filed on February 28, 2003 (Attorney Docket No. BEAS-01330US0 SRM/DTX); and

U.S. Patent Application No. _____ entitled SYSTEMS AND METHODS FOR STREAMING XPATH QUERY, by Patrick Calahan, filed February 17, 2004 (Attorney Docket No. BEAS-01330US1 SRM/DTX).

FIELD OF THE INVENTION

The present invention relates to the querying of data, such as from a document or file.

BACKGROUND

XPath is a W3C language standard that can be used to address or query parts of an XML document. It models an XML document as a tree of nodes, which can include element nodes, attribute nodes and/or text nodes. XPath can be used to identify a subset of an XML document by matching, or determining whether a node matches a pattern, similar to how SQL can be used against a database. In the typical case, an expression written in the XPath language is evaluated against an XML document to determine which parts of the document 'match' the XPath. In order to do this, the XML document must be parsed and represented in memory. One of the standard representations of XML is the Document Object Model (DOM). DOM model presents an XML document as a

hierarchy of nodes through which one can navigate arbitrarily. This approach provides a lot of flexibility, but comes at a cost in terms of efficiency and memory use, as the entire document must be brought into memory at one time.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a diagram showing an exemplary system that can be used in accordance with one embodiment of the present invention.

Figure 2 shows an exemplary data tree that can be used with the system of Figure 1 in an embodiment.

Figure 3 is a flowchart for an exemplary process that can be used with the system of Figure 1 in an embodiment.

DETAILED DESCRIPTION

The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

Systems and methods in accordance with one embodiment of the present invention overcome deficiencies in existing XML query systems by representing the XML document as a stream of discrete 'events', with each event representing a portion of the document as the document is being parsed. Event matching can be performed against the event stream. Matching events can then be routed for processing by appropriate objects or components and returned to the event stream if necessary.

XPath can be used to identify a subset of an XML document, similar to how SQL can be used against a database. XPath is a W3C language standard that can be used to address or query parts of an XML document. It can address parts of an XML document by providing basic facilities for manipulating strings, numbers, and Boolean variables. XPath operates on the hierarchical structure, which can be but is not limited to a tree, instead of the syntax of an XML document and can be used for matching, or determining whether a node matches a pattern. It models an XML document as a tree of nodes, which can include element nodes, attribute nodes and/or text nodes and defines a way to compute a string-value for each node type. The primary syntactic construct in XPath is the expression. An expression is evaluated to yield an object of type node-set,

Boolean, number, or string. In the typical case, an expression written in the XPath language is evaluated against an XML document to determine which parts of the document 'match' the XPath. In order to do this, the XML document must be parsed and represented in memory.

Systems and methods in accordance with one embodiment of the present invention adopt a true streaming approach, passing bits of an XML document one after another, and it is up to the system to decide what to do with each bit as it passes on the stream. An advantage of a true streaming approach is that such a system is faster and far more memory efficient than a DOM-style approach, since only one portion of the document is in memory at any given time. When using a streaming parser, a system can take a stream on an XML document, generating a stream of events, one event for each node in the XML tree, and perform XPath matching on that stream. A streaming XPath system can also be schema aware, such that the system knows the XML schema for a document, that schema can be used to provide insight on how to most effectively process the document. For instance, the need to go "backwards" in a stream can be avoided if the system knows in advance which events it needs to grab and in what order those events will be received.

A streaming approach can place a greater burden on a system to maintain relevant state than a DOM approach, as a streaming approach may provide no navigation mechanisms. While such an approach provides a very efficient way to process an XML document, the efficiency comes at a cost, as there can be considerably less context available when working with a stream than when working with a DOM tree. Further, XPath has to be able to traverse the hierarchy, in some sense, in order to locate the appropriate portion of the document. In many instances, it is simple to locate an appropriate portion of XML against a DOM tree, since the system is able to walk against the tree. When using a stream, a system has to maintain context in a way that is efficient enough to make using the stream worthwhile. Some tradeoffs can be made, such as not supporting the entire XPath specification. At some point, it may be more efficient to realize an entire DOM tree, if doing a convoluted matching against the entire document.

The XPath specification defines the notion of a context, where a context is the information about an event, consisting of a node it represents, a position of the node relative to a parent node, and a function library, as well as any of several other components such as variable bindings. A location path is a type of expression that can select a set of nodes relative to the context node. The evaluation of a location path

expression can result in the node-set containing the nodes being selected by the location path. Location paths can recursively contain expressions used to filter node sets. Expressions can be parsed by first dividing the character string to be parsed into tokens, then parsing the resulting token sequence.

In one embodiment, it is relatively easy to map context to the stream, as the system can maintain a stack of stream events that provide the direct ancestral line back to the root. For instance, matching an XPath that consists solely of child axes can be straightforward. In another embodiment, mapping can become more complicated in the case of descendant axes, similar to matching an entire sub-tree. In those cases, it can be necessary to spawn a tree of contexts and perform matching against each of those contexts. It can become complicated, as the system gets to maintain, and know when you can discard those cloned contexts. It can be even more complicated when matching axes called "following," which match everything below a certain point in the document. In some cases, it is necessary to maintain that context tree and track what to add on to the tree as the system navigates its way back out of the document.

Systems and methods in accordance with one embodiment of the present invention know how to manage the multi-context mode discussed in the proceeding paragraph. They utilize the information of contexts in the stack matching against the expression to recognize when to go into this multi-context mode, when to destroy those contexts, and how to update the context stack appropriately. Certain optimizations can also be used that can know when not to match certain contexts in the context tree. XPath defines different ways to slice up a document, such as parents and children, that each has to be dealt with in a different way.

Systems and methods in accordance with one embodiment do not account for reverse axes. A reverse axis is any axis that would require going "back" through the stream. A diagram showing an exemplary "forward" and "backward" or "reverse" path through a data tree is given by **Figure 2**. A diagram of an exemplary system is shown in **Figure 1**. A streaming parser **102** generates events by parsing an XML document **100**, and then places those events on an XML event stream. Such a streaming process is demonstrated by the diagram of **Figure 3**. The streaming parser first takes a tree of an XML document as the input **300**, traverses the XML tree either through a broad-first search or a depth-first search and adds each node visited into a data structure, e.g., a queue **302**. The streaming parser then processes the queue in the first-in-first-out (FIFO) manner **304** to generate an event for the context of each node in the queue **306** and

appends each event to the output stream **308**. Using the event stream, the end user of the streaming API pulls events from the stream as they come through it. When a user calls for the next event on the stream, that user has a guarantee that they will get the next event. The user will find out if the next event is going to match, and will find out before the call to next returns.

In one embodiment, an XPath matching component **104** performs matching on each event received on the stream. Matching can be communicated to a caller or end user in a number of ways. These systems are doing event-based processing, as opposed to static tree-based processing. In a tree-based implementation, for example, a user can request all the nodes that match an XPath for a document. The user will receive a collection of nodes that match that XPath. Such an approach is not necessarily effective in the case of streaming, as it is then necessary to read through the document, save all the nodes, and present the collection to the user. This is fundamentally not a stream-centric way of looking at the problem. Instead, using an XPath matching approach, an observer **106** can be registered. The registered observer is an object to be notified whenever an event comes through the stream that matches this XPath. If an event matches an XPath, that event can be temporarily diverted and sent over to a user-defined object 108 that reacts to the match. Then, the event can be returned to the stream if necessary so that any subsequent object pulling events from the stream can process that event.

One embodiment may be implemented using a conventional general purpose or a specialized digital computer or microprocessor(s) programmed according to the teachings of the present disclosure, as will be apparent to those skilled in the computer art. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art. The invention may also be implemented by the preparation of integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

One embodiment includes a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the features presented herein. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, micro drive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards,

nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

Stored on any one of the computer readable medium (media), the present invention includes software for controlling both the hardware of the general purpose/specialized computer or microprocessor, and for enabling the computer or microprocessor to interact with a human user or other mechanism utilizing the results of the present invention. Such software may include, but is not limited to, device drivers, operating systems, execution environments/containers, and applications.

The foregoing description of the preferred embodiments of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. Embodiments were chosen and described in order to best describe the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention, the various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

CLAIMS

What is claimed is:

1. A system to process an XML document, comprising:

a streaming parser capable of parsing an XML document and generating a stream of at least one event, wherein each event can represent a portion of the document;

a matching component capable of performing matching on an event in the stream and notifying an observer if the event is a match;

said observer capable of listening for a matching event and passing it to a user object; and

said user object capable of handling the matching event.

2. The system according to claim 1, wherein:

the XML document is represented in a hierarchical structure.

3. The system according to claim 2, wherein:

the hierarchical structure can be a tree with each node containing a portion of the document.

4. The system according to claim 3, wherein:

the streaming parser is capable of performing a method, comprising:

traversing the XML tree and adding visited nodes into a data structure;

processing the nodes in the data structure and generating an event for each node; and

appending the event to the output stream.

5. The system according to claim 4, wherein:

the tree can be traversed using a breath-first or depth-first search.

6. The system according to claim 4, wherein:

the data structure can be a queue.

7. The system according to claim 4, wherein:

the data structure can be processed using a first-in-first-out approach.

8. The system according to claim 1, wherein:

the matching component is capable of keeping only a portion of the XML document in memory at any given time.

9. The system according to claim 1, wherein:

the matching component is capable of knowing the schema of the XML document and foreseeing the coming events.

10. The system according to claim 1, wherein:

the matching component is capable of performing an expression-based match, which can be an XPath query.

11. The system according to claim 3, wherein:

the matching component is capable of keeping, cloning and destroying the entirety or a portion of the sub-tree descending from a node in the tree.

12. The system according to claim 1, wherein:

the user object is capable of returning the matching event to an XML stream for use by any other component.

13. A method for processing an XML document, comprising:

parsing an XML document and generating a stream of at least one event, wherein each event can represent a portion of the document;

performing matching on an event in the stream and notifying an observer if the event is a match;

listening for a matching event and passing it to a user object; and

handling the matching event.

14. The method according to claim 13, further comprising:

representing the XML document in a hierarchical structure, which can be a tree with each node containing a portion of the document.

15. The method according to claim 14, wherein:

the parsing of the XML document comprises the steps of:

traversing the XML tree and adding visited nodes into a data structure;

processing the nodes in the data structure and generating an event for each node; and

appending the event to the output stream.

16. The method according to claim 15, wherein:

the XML tree is traversed using a breath-first or depth-first search.

17. The method according to claim 15, wherein:

the data structure is processed using a first-in-first-out approach.

18. The method according to claim 13, further comprising:

keeping only a portion of the XML document in memory at any given time.

19. The method according to claim 13, further comprising:

knowing the schema of the XML document and foreseeing the coming events.

20. The method according to claim 13, further comprising:

performing an expression-based match, which can be an XPath query.

21. The method according to claim 14, further comprising:

keeping, cloning and destroying the entirety or a portion of the sub-tree descending from a node in the tree.

22. The method according to claim 13, further comprising:

returning the matching event to an XML stream for use by any other component.

23. A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

parse an XML document and generate a stream of at least one event, wherein each event can represent a portion of the document;

perform matching on an event in the stream and notify an observer if the event is a match;

listen for a matching event and pass it to a user object; and

handle the matching event.

24. The machine readable medium of claim 23, further comprising instructions that when executed cause the system to:

represent the XML document in a hierarchical structure, which can be a tree with each node containing a portion of the document.

25. The machine readable medium of claim 24, wherein the instructions that when executed cause the system to:

parse the XML document, comprising the steps of:

traversing the XML tree and adding visited nodes into a data structure;

processing the nodes in the data structure and generating an event for each node; and

appending the event to the output stream.

26. The machine readable medium of claim 25, wherein the instructions that when executed cause the system to:

traverse the tree using a breath-first or depth-first search.

27. The machine readable medium of claim 25, wherein the instructions that when executed cause the system to:

process the data structure using a first-in-first-out approach.

28. The machine readable medium of claim 23, further comprising instructions that when executed cause the system to:

perform an expression-based match, which can be an XPath query.

29. The machine readable medium of claim 23, further comprising instructions that when executed cause the system to:

keep only a portion of the XML document in memory at any given time.

30. The machine readable medium of claim 23, further comprising instructions that when executed cause the system to:

know the schema of the XML document and foresee the coming events.

31. The machine readable medium of claim 24, further comprising instructions that when executed cause the system to:

keep, clone and destroy the entirety or a portion of the sub-tree descending from a node in the tree.

32. The machine readable medium of claim 23, further comprising instructions that when executed cause the system to:

return the matching event to an XML stream for use by any other component.

33. A system for processing an XML document, comprising:

means for parsing an XML document and generating a stream of at least one event, wherein each event can represent a portion of the document;

means for performing matching on an event in the stream and notifying an observer if the event is a match;

means for listening for a matching event and passing it to a user object; and

means for handling the matching event.

34. A computer data signal embodied in a transmission medium, comprising:

a code segment including instructions to parse an XML document and generate a stream of at least one event, wherein each event can represent a portion of the document;

a code segment including instructions to perform matching on an event in the stream and notify an observer if the event is a match;

a code segment including instructions to listen for a matching event and pass it to a user object; and

a code segment including instructions to handle the matching event.

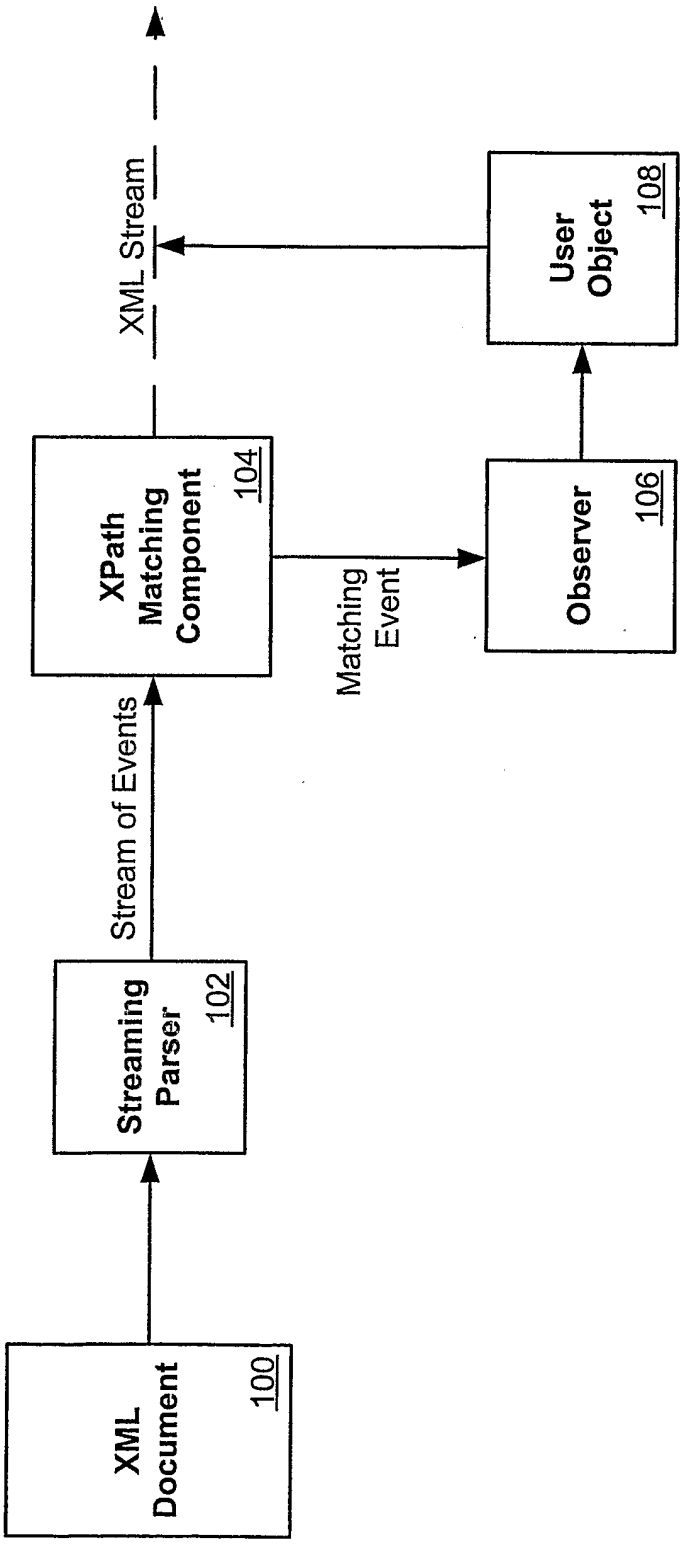

XML Document
100
Streaming Parser
102
Stream of Events
XPath Matching Component
104
Matching Event
Observer
106
User Object
108
XML Stream

Figure 1

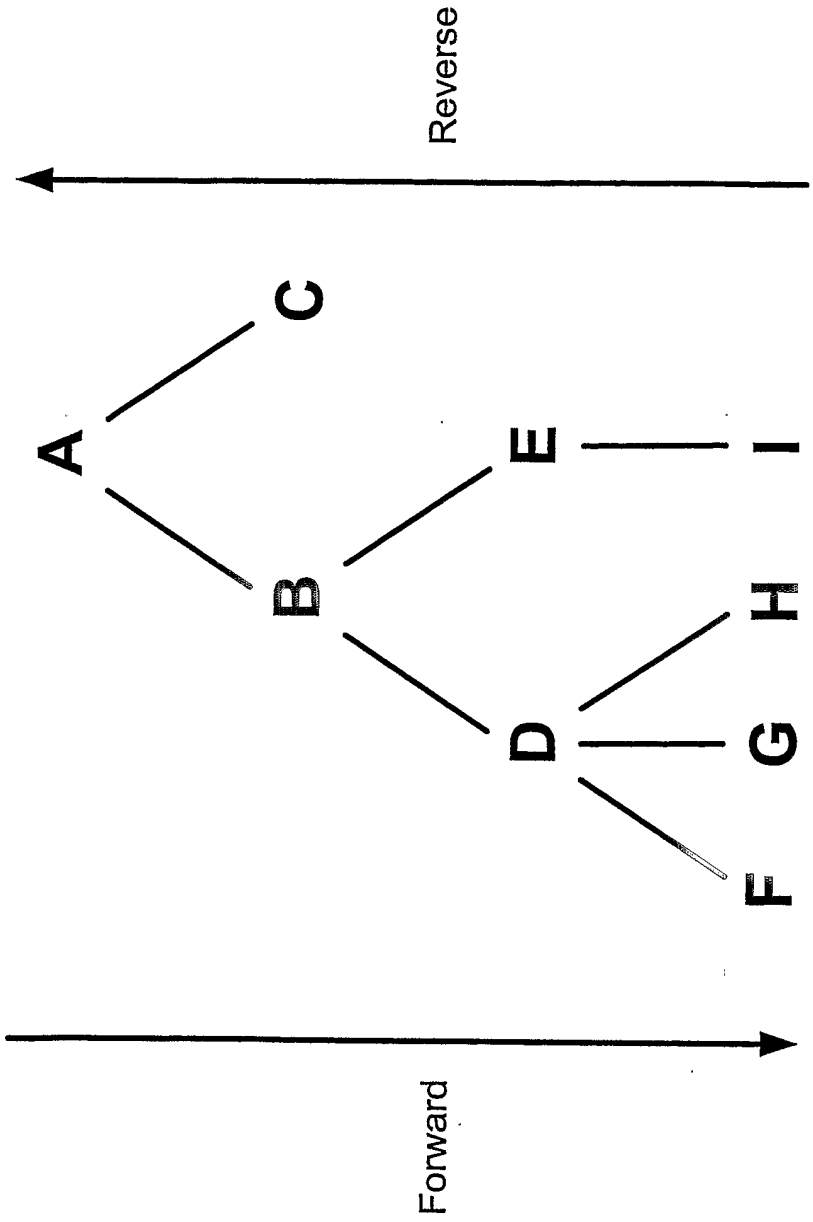


Figure 2

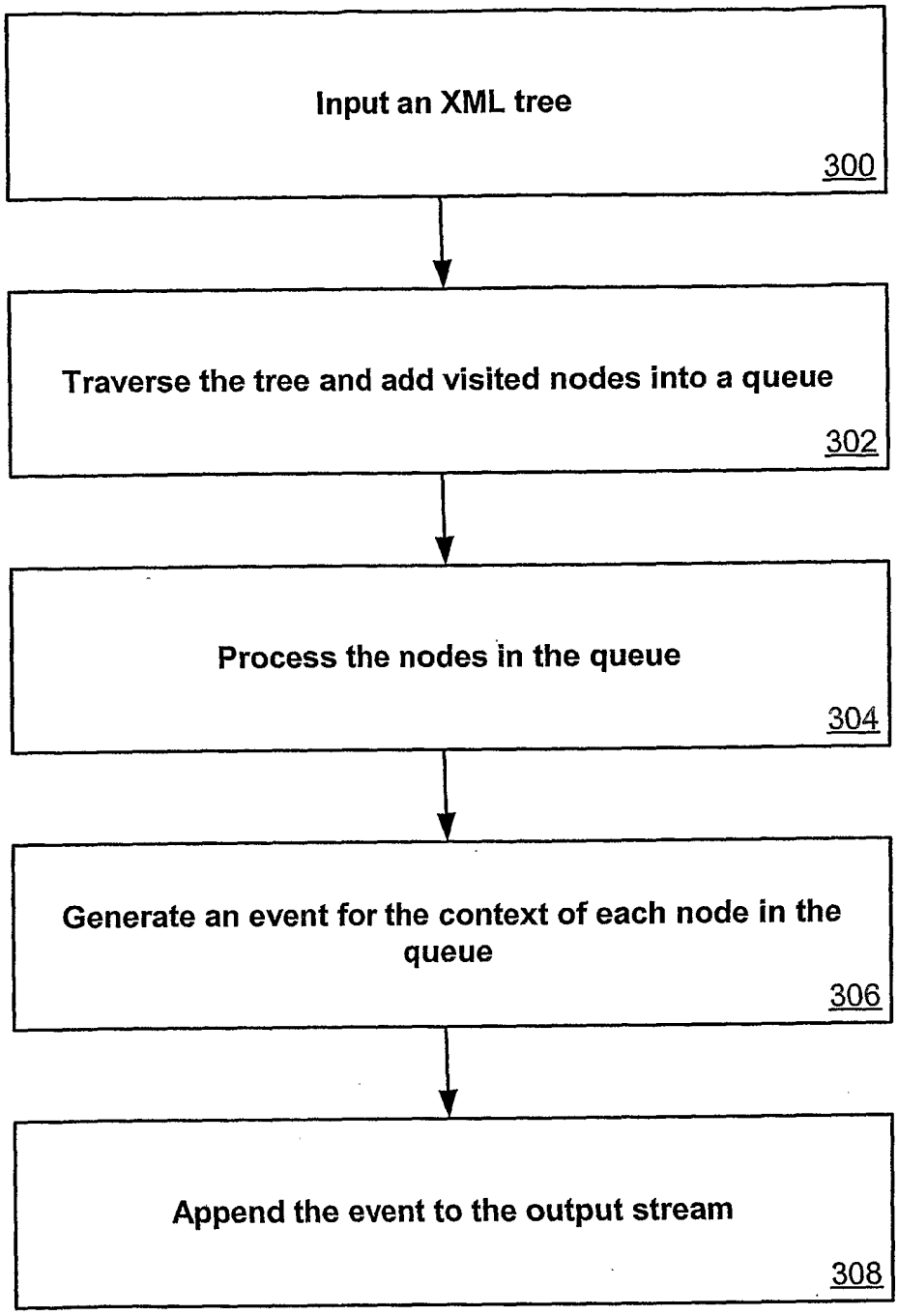

Input an XML tree
300
Traverse the tree and add visited nodes into a queue
302
Process the nodes in the queue
304
Generate an event for the context of each node in the queue
306
Append the event to the output stream
308

Figure 3