US 20050246304A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2005/0246304 A1

Knight et al. (43) Pub. Date: Nov. 3, 2005

(54) END-USER APPLICATION CUSTOMIZATION USING RULES

(75) Inventors: **Holly Knight**, Woodinville, WA (US); **Praveen Seshadri**, Bellevue, WA (US); **Robert H. Gerber**, Bellevue, WA (US); **Stephen E. Dossick**, Redmond, WA (US); **Vincent H. Curley**, Bellevue, WA (US); **Shyamalan Pather**, Seattle, WA (US)

Correspondence Address:
AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114 (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA

(21) Appl. No.: **10/903,762**

(22) Filed: **Jul. 30, 2004**

Related U.S. Application Data

(60) Provisional application No. 60/567,153, filed on Apr. 30, 2004. Provisional application No. 60/567,149, filed on Apr. 30, 2004. Provisional application No. 60/567,165, filed on Apr. 30, 2004.

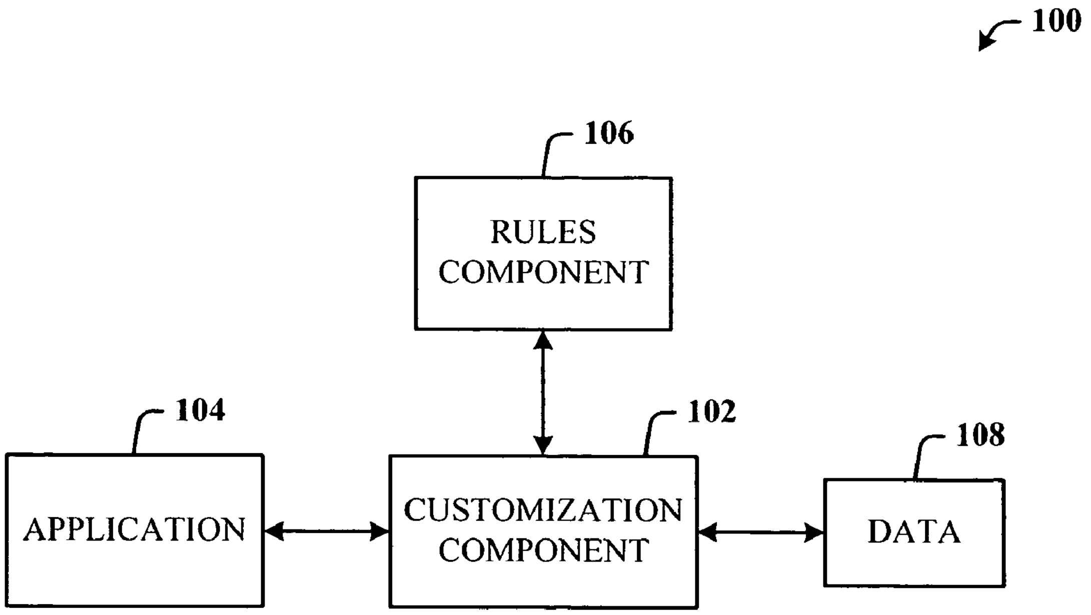
Publication Classification

(51) **Int. Cl.**^{7} **G06F 17/00**; G06F 7/00; G06N 5/02; G06F 9/44

(52) **U.S. Cl.** **706/47**; 717/110

(57) **ABSTRACT**

Rules architecture that facilitates data management of an application such that the application can be personalized by the end-user for the end-user. Included is a customization component that facilitates the exposing of an application generated event to an end-user. A rules component allows the end-user to create one or more rules to process the event, which one or more rules facilitate the submission of application data associated with the event for external and internal processing.



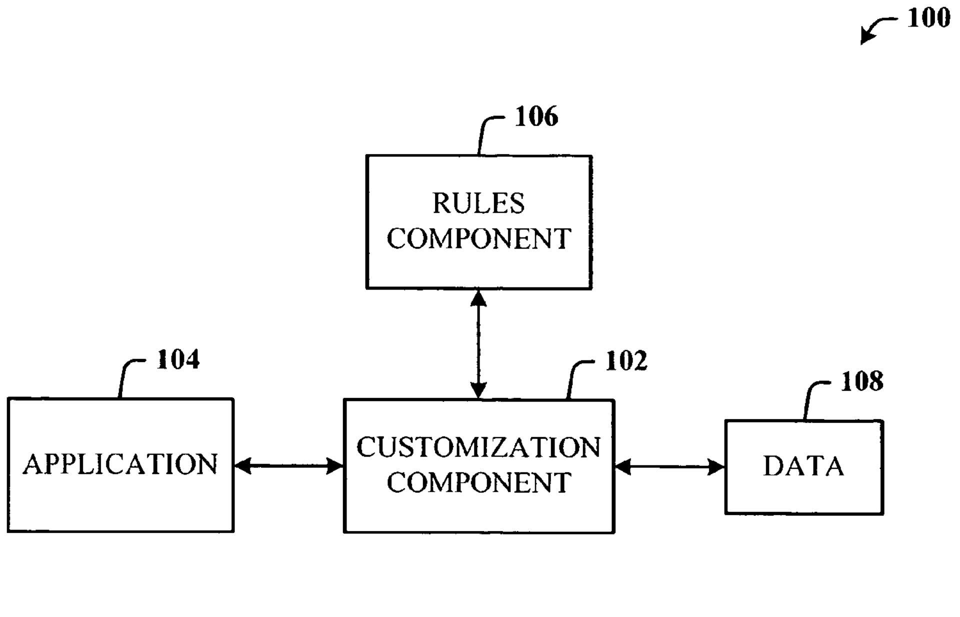

100
106
RULES COMPONENT
104
APPLICATION
102
CUSTOMIZATION COMPONENT
108
DATA

FIG. 1

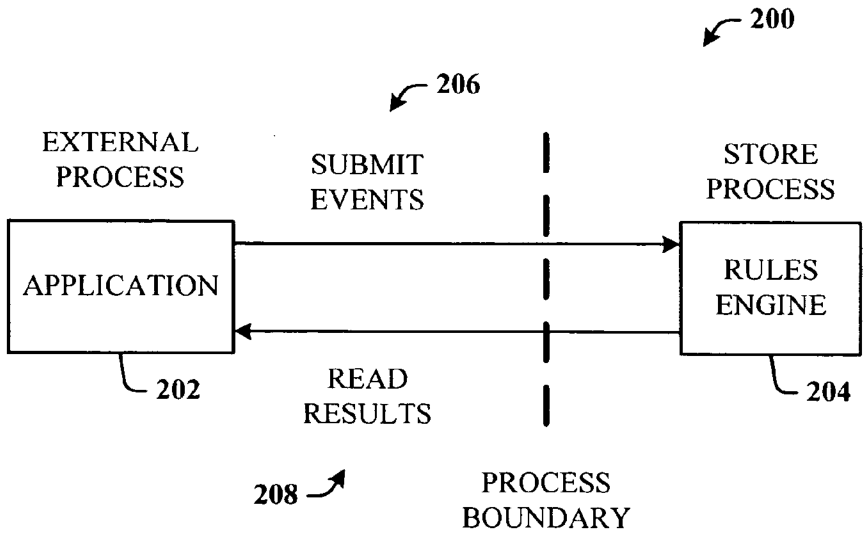

200
206
EXTERNAL PROCESS
SUBMIT EVENTS
STORE PROCESS
APPLICATION
RULES ENGINE
202
READ RESULTS
204
208
PROCESS BOUNDARY

FIG. 2

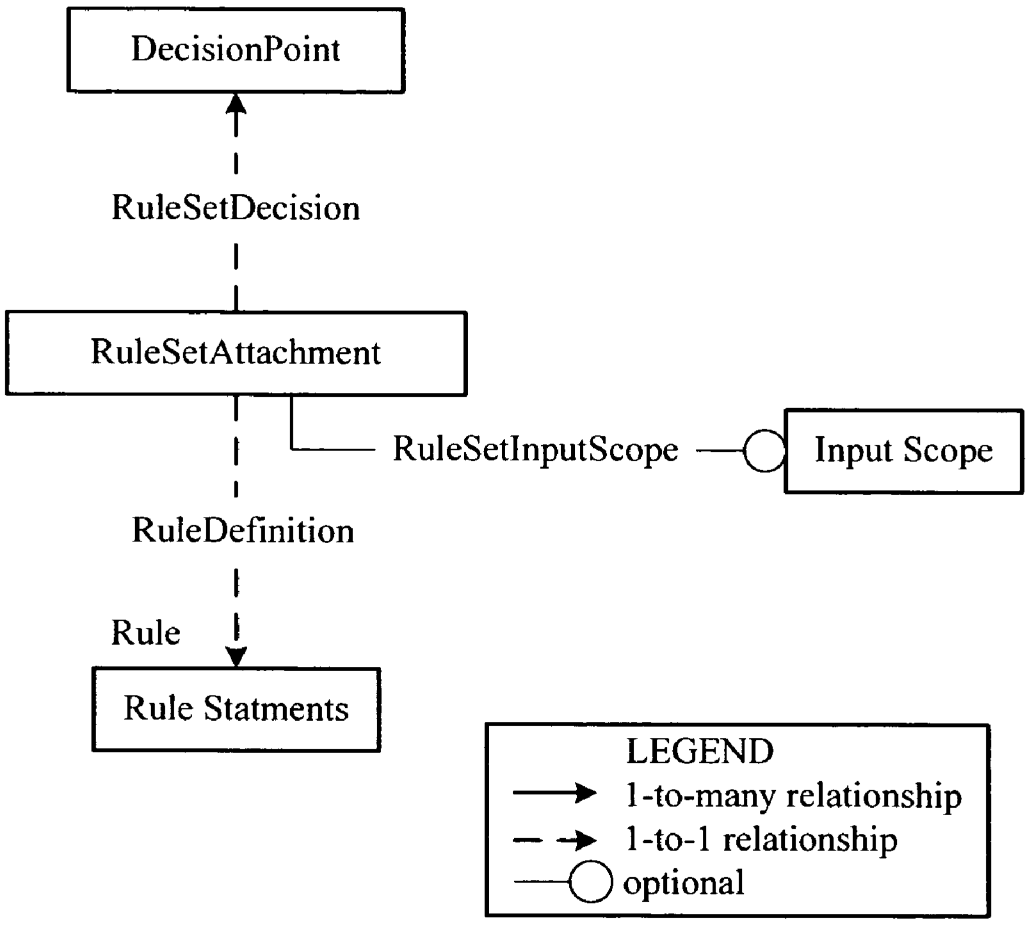

DecisionPoint
RuleSetDecision
RuleSetAttachment
RuleSetInputScope
Input Scope
RuleDefinition
Rule
Rule Statments
LEGEND
1-to-many relationship
1-to-1 relationship
optional

FIG. 3

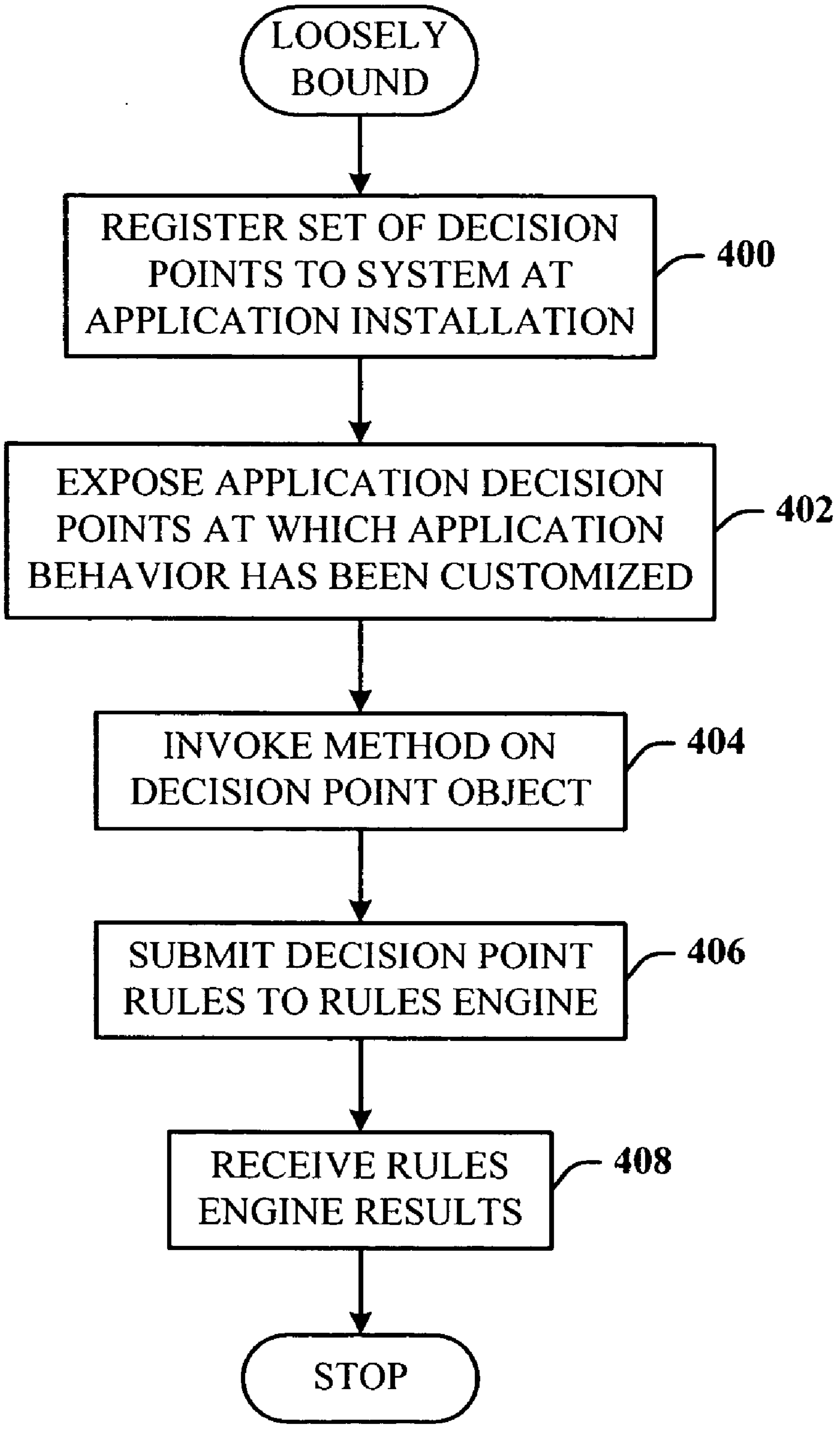

LOOSELY BOUND
REGISTER SET OF DECISION POINTS TO SYSTEM AT APPLICATION INSTALLATION
400
EXPOSE APPLICATION DECISION POINTS AT WHICH APPLICATION BEHAVIOR HAS BEEN CUSTOMIZED
402
INVOKE METHOD ON DECISION POINT OBJECT
404
SUBMIT DECISION POINT RULES TO RULES ENGINE
406
RECEIVE RULES ENGINE RESULTS
408
STOP

FIG. 4

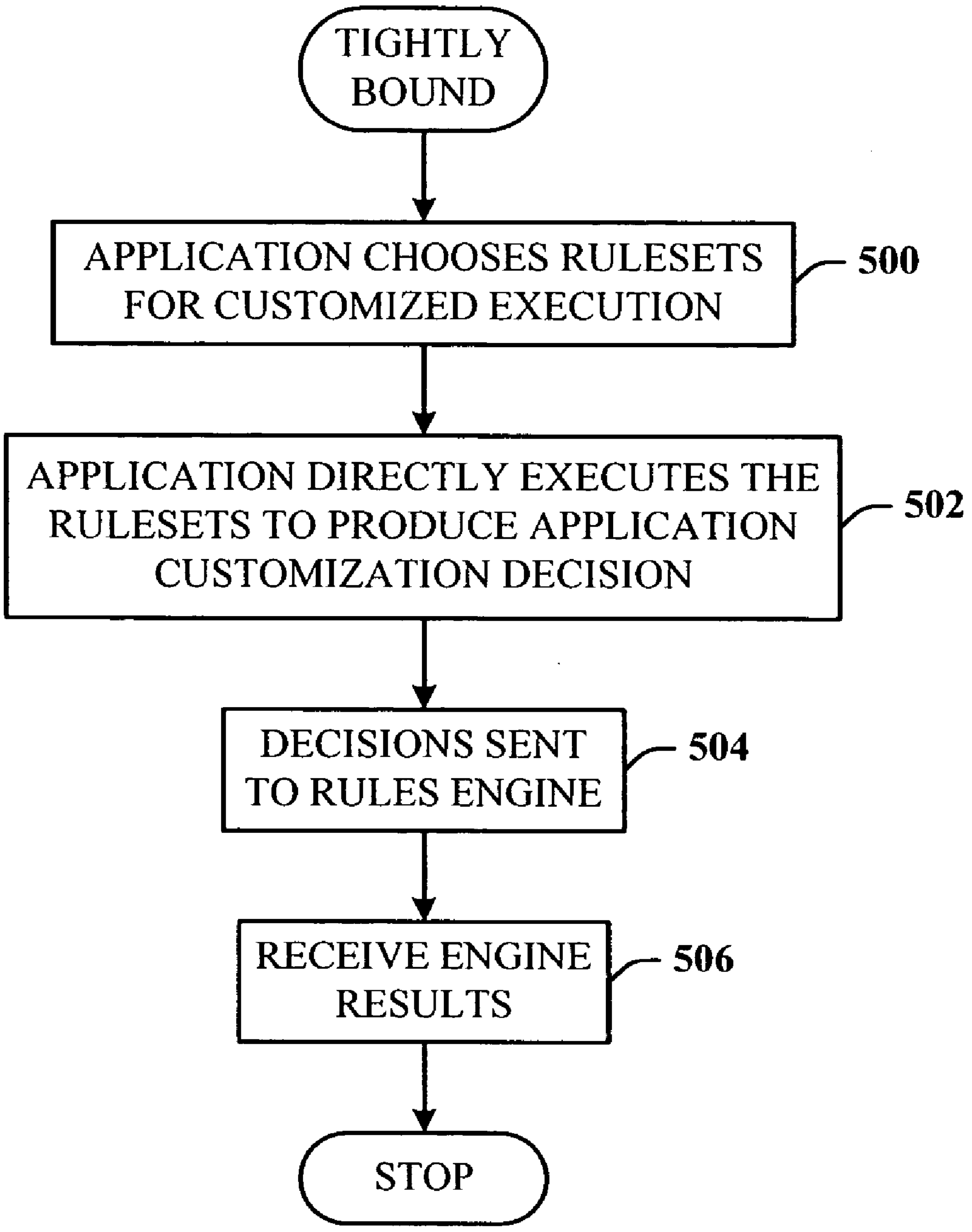

TIGHTLY BOUND
APPLICATION CHOOSES RULESETS FOR CUSTOMIZED EXECUTION
500
APPLICATION DIRECTLY EXECUTES THE RULESETS TO PRODUCE APPLICATION CUSTOMIZATION DECISION
502
DECISIONS SENT TO RULES ENGINE
504
RECEIVE ENGINE RESULTS
506
STOP

FIG. 5

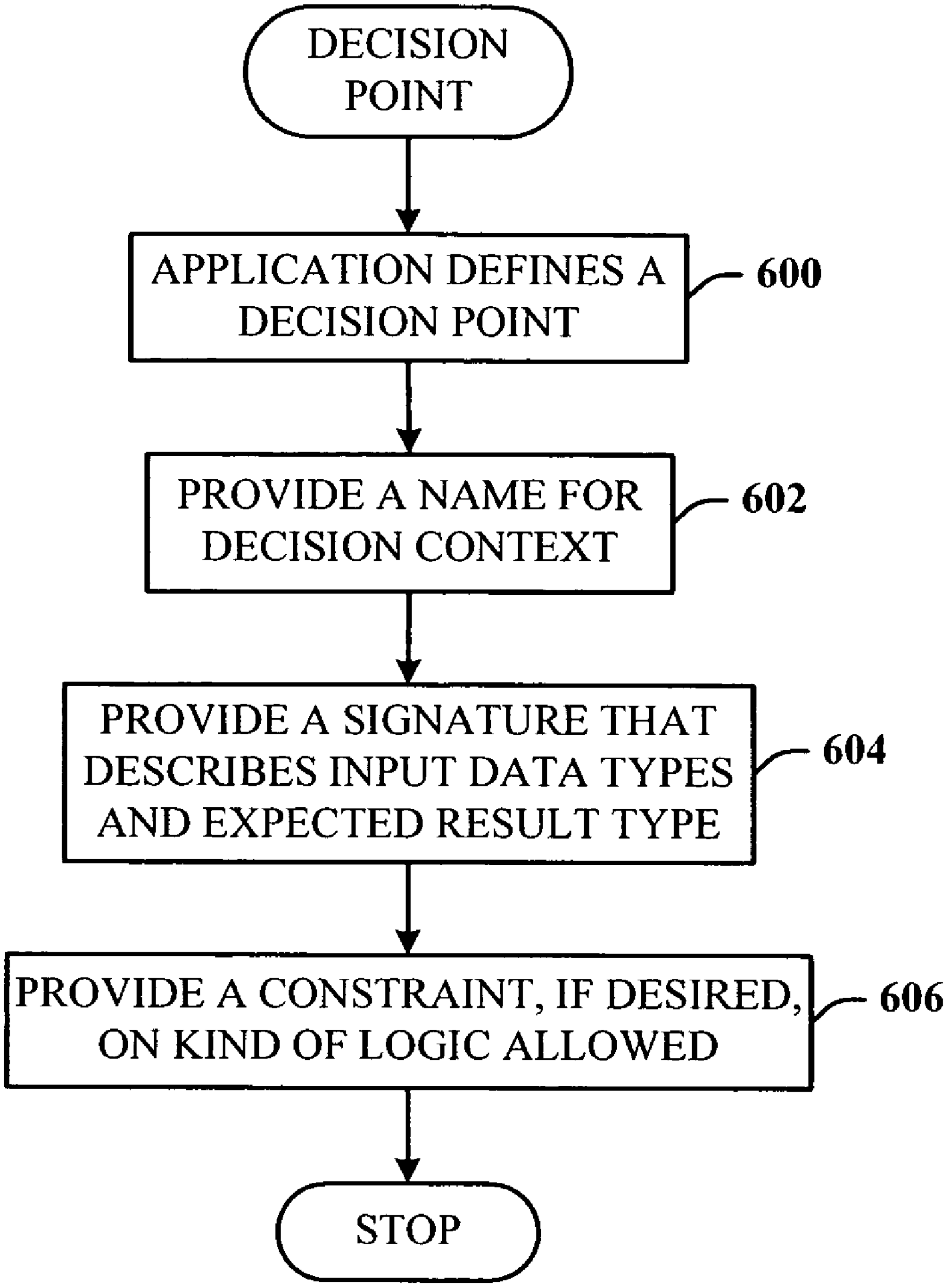

DECISION POINT
APPLICATION DEFINES A DECISION POINT
600
PROVIDE A NAME FOR DECISION CONTEXT
602
PROVIDE A SIGNATURE THAT DESCRIBES INPUT DATA TYPES AND EXPECTED RESULT TYPE
604
PROVIDE A CONSTRAINT, IF DESIRED, ON KIND OF LOGIC ALLOWED
606
STOP

FIG. 6

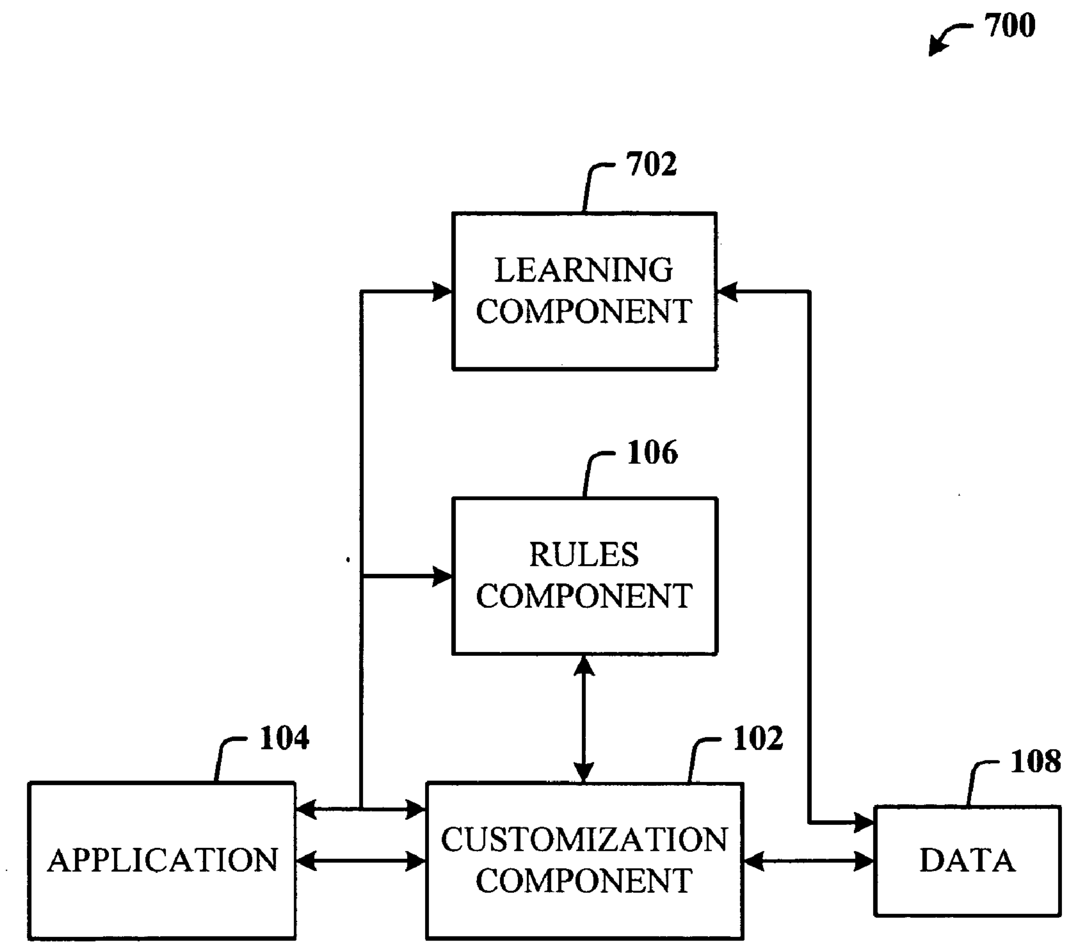

700
702
LEARNING COMPONENT
106
RULES COMPONENT
104
APPLICATION
102
CUSTOMIZATION COMPONENT
108
DATA

FIG. 7

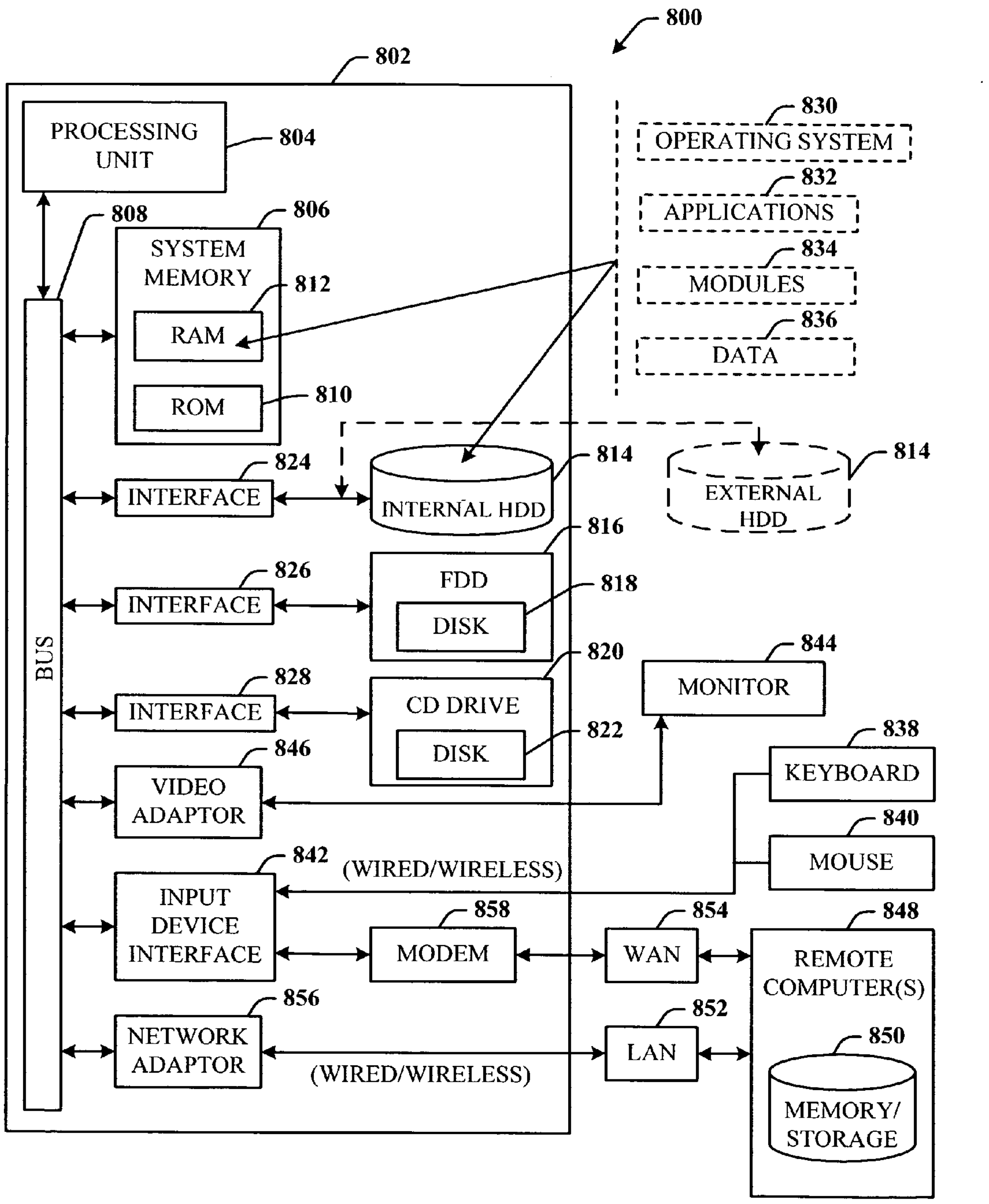

800
802
PROCESSING UNIT
804
808
806
SYSTEM MEMORY
812
RAM
ROM
810
BUS
830
OPERATING SYSTEM
832
APPLICATIONS
834
MODULES
836
DATA
824
INTERFACE
INTERNAL HDD
814
EXTERNAL HDD
814
816
826
INTERFACE
FDD
DISK
818
820
828
INTERFACE
CD DRIVE
DISK
822
844
MONITOR
846
VIDEO ADAPTOR
838
KEYBOARD
840
MOUSE
842
(WIRED/WIRELESS)
INPUT DEVICE INTERFACE
858
MODEM
854
WAN
848
REMOTE COMPUTER(S)
856
NETWORK ADAPTOR
852
LAN
850
MEMORY/ STORAGE
(WIRED/WIRELESS)

FIG. 8

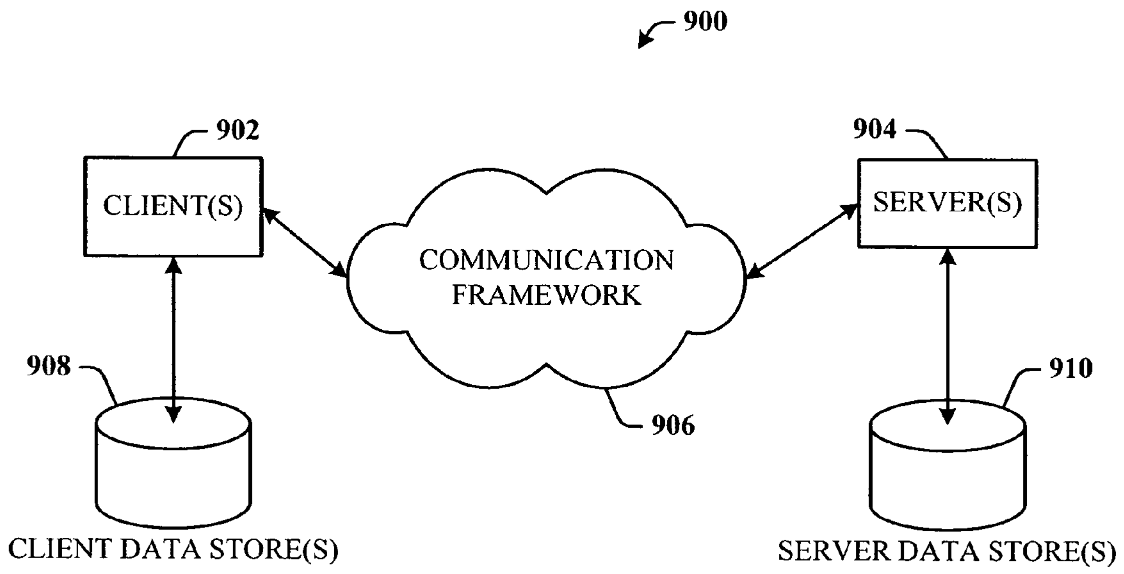

900
902
CLIENT(S)
904
SERVER(S)
COMMUNICATION FRAMEWORK
906
908
910
CLIENT DATA STORE(S)
SERVER DATA STORE(S)

FIG. 9

END-USER APPLICATION CUSTOMIZATION USING RULES

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application clams the benefit of U.S. Provisional Patent Application Ser. No. 60/567,153 entitled "END-USER APPLICATION CUSTOMIZATION USING RULES", filed on Apr. 30, 2004, U.S. Provisional Patent Application Serial No. 60/567,149 entitled "DERIVED SET—A RULES-BASED QUERY-LIKE MECHANISM THAT DEFINES CONTENTS OF A COLLECTION" filed on Apr. 30, 2004, and U.S. Provisional Patent Application Ser. No. 60/567,165, entitled "RULES FRAMEWORK FOR DEFINITION AND EXECUTION OF END-USER RULES LOGIC", filed on Apr. 30, 2004. This application is also related to co-pending U.S. Patent Application Ser. No. ______ (Atty. Dkt. No. MSFTP668USA) entitled "RULES FRAMEWORK FOR DEFINITION AND EXECUTION OF END-USER RULES LOGIC" filed on Jul. 30, 2004. The entireties of the above-noted applications are incorporated herein by reference.

TECHNICAL FIELD

[0002] This invention is related to software that facilitates application customization, and more specifically, a rules abstraction architecture that facilitates application customization by an end-user.

BACKGROUND OF THE INVENTION

[0003] Computers and computing have always divided the world of users into two classes: the knowledgeable "high priests" who know how to use computers in complex ways, to shape programs and enable behaviors that are valuable and rich; and the novice users, who are at their mercy, denied easy or cheap access to knowledge or information or the education to make computers serve their needs well. However, major breakthroughs in computing have occurred when technology has broken down some of these barriers to access.

[0004] In the world of the mainframe, computers were too expensive for all but the largest businesses to afford. The advent of mini-computers, and then personal computers (PCs), broke down the cost barrier and made computers available to small businesses and individuals.

[0005] In the 1980's, programmers struggled to build graphical user interface (GUI) applications, and without rich and consistent GUIs, were unable to build valuable applications for PC users. The Visual Basic revolution and the use of controls and event-based GUI construction enabled a whole army of application developers who could easily build rich applications. This also established a virtuous cycle with many more end-users who could exploit these applications.

[0006] In the 1990's, end-users struggled to overcome a lack of access to information. The growth of the Internet and the web transformed this space, making almost all valuable information accessible to anyone with a browser. However, there are still significant barriers to overcome.

[0007] Computing is not personal. There is very little about a PC that is truly "personal". The data on the local disk is personal. But the behavior of the machine (what it does on behalf of the user) is close to identical across millions of users. Despite owning an amazingly powerful general purpose computer, the average user treats it as a static tool, useful as a communication end-point, useful as a search entry-point, useful to execute some canned mass-market applications, but otherwise incapable of any "personal computing" in the true sense of the word. The personalization capabilities available in current applications just scratch the surface of what is possible and desirable.

[0008] Computing is manual. Consider the daily routine of most typical computer end-users. The PC gathers information, reacts to communications, makes decisions and acts upon them—initiates or responds to communications, organizes information, buys and sells goods, travel, etc. Computers have improved communication between people, and have improved access to information. However, PCs have done little to relieve the end-user's responsibility to make decisions and act upon them at the right time. In the business world, there are decision support systems for major organizational decisions. Still software does not help the average PC user in the many everyday, yet important and personal decisions.

[0009] Computing is not contextual. Computer software typically provides optional settings that are rather static and unrelated to the actual context of the user (e.g., "Why should I have the same set of messaging buddies at work and at home?").

[0010] Thus, users are still in the "pre-industrial age" of software by being increasingly trapped in the tyranny of manual information processing—spending hours every day sifting, sorting, searching, and reacting to e-mail, documents and other personal data.

[0011] End-user software should be personalized, aware of the needs and preferences of the end-user, acting specifically in a manner guided by those needs and by the user context. Further, computer systems and software should provide every end-user with a personal executive assistant who works 24 hours a day gathering and sifting information of interest to the end-user, and reacting to that information.

[0012] The most valuable class of end-user computing activity deals with information flows and search, such as ensuring the end-user sees relevant information (e.g., "Tell me if there is a school closing due to bad weather."), enhancing person-to-person communication with personalized behavior (e.g., "If I'm out of the office when my wife calls, let her know when I'll be back."), ensuring important information is not lost (e.g., "If there's an urgent e-mail, make sure it gets forwarded to me on my mobile device."), and automating the management of information (e.g., "As new photographs arrive, place them in the right folders and shares based on their timestamp, GPS location, and any relevant calendar entries.").

[0013] The way to accomplish this is by allowing the end-user to "program" the behavior of the computer. However, traditional programming languages are clearly not the answer, in that, the end-user is not (and cannot become) a trained developer.

[0014] What is needed is an improved mechanism that allows an end-user to personalize an operating system and an application.

SUMMARY OF THE INVENTION

[0015] The following presents a simplified summary of the invention in order to provide a basic understanding of some aspects of the invention. This summary is not an extensive overview of the invention. It is not intended to identify key/critical elements of the invention or to delineate the scope of the invention. Its sole purpose is to present some concepts of the invention in a simplified form as a prelude to the more detailed description that is presented later.

[0016] The present invention disclosed and claimed herein, in one aspect thereof, comprises architecture that facilitates data management of an application. Applications can now be personalized by the end-user for the end-user. Included is a customization component that facilitates the exposing of an application generated event to an end-user; and a rules component that allows the end-user to create one or more rules to associate with the event, which one or more rules facilitate the submission of application data associated with the event for external processing. Since each rule-enabled feature of the application is defined by the decision point, an event/trigger/change at the decision point can be managed by the end-user for various purposes. Application data exposed by the decision point can be processed externally to return a result to the application that modifies behavior of the application.

[0017] Various functions of the application are rule-enabled. The end-user can then create rules for those functions to further manipulate data associated with the functions. Two kinds of application customization are called loosely-bound and tightly bound customization. In a loosely-bound customization, the application exposes an interceptor point, or "decision point" at which the end-user can attach customization rules. A rules is associated with the decision point by setting up an attachment type. Multiple different rules can be attached to a single application customization decision point using corresponding attachment types. The application does not "know" about the attached rules, but calls a method on the decision point item, and then processes the results. The method call causes a rules engine to be invoked on the right rules.

[0018] In a tightly-bound customization, decision points are not used. The application invokes the rules directly, which allows greater control to the application on which the rules will be invoked, and what types of rules are allowed when the rule engine processes rules to return application customization results. Thus, the tightly-bound application identifies the correct rules for the rules-based decision and user by issuing a file system query for the rules that correspond to that application, user, decision and input, to be provided with the rules. In contrast with the loosely-bound application, the choice of rules based on the user, decision, and input is determined by the rules platform whenever the desision point input method is invoked by the calling application.

[0019] In another aspect of the present invention, a learning component is provided that faclitates the application learning end-user behavior, and captures the behavior in the form of rules.

[0020] To the accomplishment of the foregoing and related ends, certain illustrative aspects of the invention are described herein in connection with the following description and the annexed drawings. These aspects are indicative, however, of but a few of the various ways in which the principles of the invention can be employed and the present invention is intended to include all such aspects and their equivalents. Other advantages and novel features of the invention will become apparent from the following detailed description of the invention when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0021] **FIG. 1** illustrates a system that facilitates application customization in accordance with the present invention.

[0022] **FIG. 2** illustrates a block diagram of a model for interaction of a customized application with a rules engine in accordance with the present invention.

[0023] **FIG. 3** illustrates a diagram of item types and their relationships of the rules architecture.

[0024] **FIG. 4** illustrates a flow chart of application customization of a loosely bound model from a runtime perspective in accordance with the present invention.

[0025] **FIG. 5** illustrates a flow chart of a methodology for a tightly bound model from a runtime perspective in accordance with the present invention.

[0026] **FIG. 6** illustrates a flow chart of one methodology associated with instantiation of a decision point in accordance with the present invention.

[0027] **FIG. 7** illustrates a system that employs a learning component in accordance with the present invention.

[0028] **FIG. 8** illustrates a block diagram of a computer operable to execute the disclosed architecture.

[0029] **FIG. 9** illustrates a schematic block diagram of an exemplary computing environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0030] The present invention is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It may be evident, however, that the present invention can be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the present invention.

[0031] As used in this application, the terms "component" and "system" are intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a server and the server can be a component. One or more components can reside within a process and/or thread of execution, and a component can be localized on one computer and/or distributed between two or more computers.

[0032] As used herein, the term to "infer" or "inference" refer generally to the process of reasoning about or inferring states of the system, environment, and/or user from a set of observations as captured via events and/or data. Inference can be employed to identify a specific context or action, or can generate a probability distribution over states, for example. The inference can be probabilistic-that is, the computation of a probability distribution over states of interest based on a consideration of data and events. Inference can also refer to techniques employed for composing higher-level events from a set of events and/or data. Such inference results in the construction of new events or actions from a set of observed events and/or stored event data, whether or not the events are correlated in close temporal proximity, and whether the events and data come from one or several event and data sources.

[0033] Rules Architecture

[0034] The rules architecture of the present invention is a platform against which developers can build applications that run rich end-user logic in the form of rules. Application customization provides the choice of a loosely-bound model, a tightly-bound model, or both, which can be employed in an application to support data driven end-user choices. An application can expose one or more triggering events. If the application is using the tightly bound model there may be no exposed triggering event—just a place in the flow of logic in the application where the application supplies input to one or more rules, gets the results, and processes those results. The rules selected are queried for by the application. In the loosely-bound model, the application itself supplies the rule triggering events (in the form of items called decision points) as input to a rules engine. By providing a decision point, the application gives the end-user the capability to control the application decision with rules. The rules engine then evaluates rule conditions and returns application specific results for those rules whose conditions evaluated to true. The results are then interpreted by the rules customized application.

[0035] Referring now to **FIG. 1**, there is illustrated a system **100** that facilitates application customization for data management in accordance with the present invention. The system **100** includes a customization component **102** that facilitates the exposing of an application generated event in an application **104** to an end-user. A rules component **106** allows the end-user to create one or more rules to associate with the event, which one or more rules facilitate the submission of application data **108** associated with the event for external processing. Various functions of the application are rule-enabled. The end-user can then create rules for those functions to further manipulate data associated with those functions. Each rule-enabled feature of the application is defined by a decision point, included as part of the customization component **102**. Thus, an event/trigger/change at the decision point can be managed by the end-user for various purposes. Application data exposed by the decision point can be processed externally to return a result to the application that modifies behavior of the application.

[0036] Referring now to **FIG. 2**, there is illustrated a block diagram of a model **200** for interaction of a customized application **202** with a rules engine **204** in accordance with the present invention. The rules architecture is a platform against which developers can build applications that run rich end-user logic in the form of rules. For application customization, the application itself supplies rule triggering events **206** as input to the engine **204**, the rules engine **204** evaluates rule conditions, and returns application specific results **208** for those rules whose conditions evaluated to true. The results are then interpreted by the rules customized application. A process boundary separates external process (e.g., the application **202**) from a store process in which the rules engines **204** runs. The application **202** submits events across the process boundary to the rules engines **204**, which then returns results back across the boundary that are run by the external processes.

[0037] The existence of the process boundary between the application **202** and the rules engine **204** is merely an implementation choice, as indicated by the dashed line. Another implementation choice can be to embed the rules engine **204** into the application **202** so that invoking it does not cause a process switch.

[0038] The rules engine **204** supports application customization. Application customization is the model that is used to support data driven end-user choices of what had traditionally been relatively fixed option settings in applications. For example, today, most e-mail authoring applications allow the user to specify a single signature file for all outgoing e-mail. An example of application customization is to employ rules that allow the user to set the signature file for outgoing e-mail based on the recipient of the e-mail.

[0039] The rules platform facilitates the use of one or both of the following models for application customization: a first model that called loosely bound, and a second model that is called tightly bound.

[0040] Referring now to **FIG. 3**, there is illustrated a diagram of item types and their relationships of the rules architecture of the present invention. An end-user logic "program" is a rule—a set of statements. The rule is the complete unit of logic. Each rule is a unit of authoring. Note that the input to the rule is a data item. The rule is a declarative statement about an item of a particular item type. As an extension to the basic model, non-item data (transient data or XML) can be provided as a rule input. The items to which the rules apply depends on the deployment of the rules. The rule is the unit of deployable end-user logic. The rules is deployed by attaching it to an item scope (a source of item inputs or decision points). This association is captured in the file system as the RuleSetAttachment (RSA). An RSA represents a connection between the DecisionPoint and Rule items. These connections can be modeled either as physical, stored "Links", or as computed "common value associations". In either case, the function is the same—the connection from a Rule to the DecisionPoint for which is was created. Rules are all file system items.

[0041] The input scope is any item of any type. Input scopes are used by the operating system to limit the scope of rule evaluation to a specific item or folder, but are typically not used by individual applications. The labels on the lines in the above graphic show the names of the relationships between items. The rule set attachment item relates a decision point to a rule (and to an input scope, if one exists). The rule set item contains relationships to zero or more rule statements. Decision point items enable an application to use the rules platform. Rules describes rule items, including constraints, conditions, and results. Rules contain rule statements. Rule statements are statements of the form: on input, if condition, then results. For application customization, the results are defined by the application and consumed by the application. Rule set attachments items store information about the connection between a decision point and a rule set. The user attaches rules in a rule set, and the application provides input to the decision point to return results.

[0042] Referring now **FIG. 4**, there is illustrated a flow chart of application customization of a loosely bound model from an runtime perspective in accordance with the present invention. While, for purposes of simplicity of explanation, the one or more methodologies shown herein, e.g., in the form of a flow chart, are shown and described as a series of acts, it is to be understood and appreciated that the present invention is not limited by the order of acts, as some acts may, in accordance with the present invention, occur in a different order and/or concurrently with other acts from that shown and described herein. For example, those skilled in the art will understand and appreciate that a methodology could alternatively be represented as a series of interrelated states or events, such as in a state diagram. Moreover, not all illustrated acts may be required to implement a methodology in accordance with the present invention.

[0043] At **400**, the application registers a set of decision points to the operating system during application installation. Decision points are created by an application as a mechanism to submit input to the rules engine. By exposing a decision point, the application gives the end-user the ability to control an application decision with rules. Input data flows to the rules engine (either directly or via a decision point) to produce application customization results. A decision point is only needed for loosely-bound application customization. At **402**, application decision points are exposed at which application behavior has been customized. At **404**, a method is invoked at each decision point object to process the end-user customization at that decision point. When the application code reaches a point at which a decision needs to be made, it invokes a method on a decision point object (item). This causes the rules engine to evaluate appropriate end-user rules and return the expected results. At **406**, the method submits the decision point rules to the rules engine. At **408**, the rules engine processes the rules and sends one or more results back to the application. The application then processes the results in an application-specific manner. The process then reaches a Stop block.

[0044] As indicated, for loosely bound application customization, the application does not know about the rules. It simply calls a method on the decision point item, and then processes the returned results. The method call causes the rules engine to be invoked on the right rules.

[0045] Referring now to **FIG. 5**, there is illustrated a flow chart of a methodology for a tightly bound model from a runtime perspective in accordance with the present invention. At **500**, the application chooses rules for customized execution. At **502**, the application directly executes the rules to produce an application customization decision. At **504**, the decision(s) are sent to the rules engine. At **506**, the rules engines processes the decisions, and returns the results back to the application, where the application processes the results in an application-specific manner. The process then reaches a Stop block.

[0046] As indicated, tightly bound applications are applications that invoke rules directly, which allows greater control to the application on which rules will be invoked, and what types of rules are allowed when the rule engine processes rules to return application customization results. This control places a greater burden on the tightly bound application. Specifically, a tightly bound application identifies the correct rules for the rules-based decision and user by issuing a query for the rules that correspond to the application and user, decision, and input to be provided to the rules. Instead of using a decision point, the application directly invokes an execute method on the rules, passing it the input item and the rule constraint. The rule constraint is used to identify the subset of rules that should be applied to the particular input.

[0047] A rule constraint can more accurately be thought of as a rule signature. The signature is used to filter which rules run given a particular invocation with a particular input. Rule constraints are specified in two places: on each rule, and on each submission point. There are no rule constraints on RSAs.

[0048] On rules, the rule constraint is used for enforcement to ensure that the rule conforms to what it says it conforms. The output of each action matches what the rule constraint says the output is. This is the same for the input. The constraint on a rule dictates where it can be attached. Validation will fail if inappropriate attachment of the rule is attempted. Rules are attached to decision points. Only rules consistent with the decision point constraints will be allowed to be attached.

[0049] To contrast this, for loosely bound applications, the choice of rules based on user, decision, and input is performed by the platform whenever the decision point input method is invoked by the calling application.

[0050] One or more rules are associated with a decision point by setting up the RSA. A particular RSA can be more restrictive than the rule constraint provided by the decision point. That is, it may specify that the rules should be applied only for a subset of the potential invocations of the decision point by the application. Specifically, it can constrain the rules along any of the following dimensions:

[0051] User: every RSA is associated with a particular user, and is only invoked for those decisions that are requested on behalf of that user.

[0052] Type: the RSA can specify that the attached rules should only apply to inputs of a particular type.

[0053] Input Scope: the RSA can, in general, specify application-specific information that causes the rules to be invoked only for a subset of potential invocations. In the case of the built-in application that allows control of changes in the item store to be controlled by the user with rules, the following kinds of input scope are expected: an item defining containment scope for query or change eventing; and, if the application supports composite operations in a single decision point (e.g., New Item or Item Modified), then the RSA can specify a further constraint which can specify a subset of the operations.

[0054] Input Content: the RSA can, in general, specify content-based filters (e.g., only apply this rule to input items owned by user). In one implementation, these filters could be deduced from the attached rule.

[0055] Note that based on this reasoning, there can be multiple RSAs attached to an application customization decision point, and the platform has to support it.

[0056] Referring now to **FIG. 6**, there is illustrated a flow chart of one methodology associated with instantiating a decision point in accordance with the present invention. At **600**, the methodology is to define each decision point. At **602**, the application provides a name. This identifies the decision context (or nature) in which the end-user rules should be applied. At **604**, the application provides a signature (or application name) that identifies the owner of the decision point in terms of input data types and the expected result type. In one implementation, transient data is also supported (via, for example, XML-eXtended Markup Language) to support cases where the data does not naturally live in the storage system, and where the overhead of persisting the data is unreasonable. At **606**, the application provides any constraint on the kind of rules logic allowed for this decision. For example, some conditions are allowed and some are not. In one implementation, the granularity of rule constraints can be limited to entire item types (i.e., a rule constraint can be expressed in terms of item types). The process then reaches a Stop block.

[0057] When a decision point is invoked with a particular input of a particular type, only those RSAs whose constraints (user, type, scope, content filter) match this input are invoked. The results from all RSAa are aggregated into a single ResultSetElement (RSE). This return value contains the aggregate results (after conflict resolution) of all attached rules.

[0058] The application should expect that all results correspond to the rule constraint defined in the decision point and it is possible for the application to distinguish between the results produced by each matching RSA.

[0059] In accordance with one optimization, a rule is allowed to have different kinds of rules, that is rules with different constraints. The effect of this is that the rules engine dynamically evaluates these rules constraints to decide which of the rules should be applied. It also means that the "direct" execution interface requires a rule constraint. If it was required to only have rules of the same constraint within it, then constraint checking can be performed at the time of rule attachment and membership. However, this reduces the flexibility of using a rule as an organizational device.

[0060] The lack of a RuleConstraint in a tightly-bound application does not mean that there is no end-user customization. The rules can be just as rich in tightly-bound, as in loosely-bound scenarios. What is missing from the tightly-bound scenario is the capability for a Universal Rules user interface (UI) to present the customization options. There is nothing for it to reflect on in order to determine those options. It is up to the tightly-bound application to present its custom UI for rules building. In this way it can control the available conditions and actions.

[0061] Rules Input/output (I/O) Model

[0062] The rules engine receives input, evaluates rules based on this input, and returns the results of any matched rules as output to the calling application. To support application customization and non-item data, the I/O model is extended to return Application Views rather than RSE. Application Views describes structured data returned from rules evaluation. The action model is extended to execute actions contained in an Application View. An RSE can be optionally persisted if the application requests it, but it will not be the expected pattern for customizing applications.

[0063] Input to the rules engine consists of an operation name and, optionally, can point to an item to which it is related. Inputs can be thought of as "verbs", like "item created", which can have a pointer to the item which has been "created". For instance, given the situation in which a new MP3 file is created in a folder, the item creation is the input operation, and the MP3 would be an item referenced by this input submission.

[0064] In addition to support for referring to an item as part of submitting input to the rules engine, it is possible for inputs to refer to XML data.

[0065] DecisionPoint Items contain methods, for example, DecisionPoint.Submit( ) and DecisionPoint.SubmitAndWait( ) are used by developers to submit input into the rules engine. Each DecisionPoint defines the operation name and (optional) item type it accepts as Input.

[0066] Submitting input that refers to existing types is thus straightforward. When an application would like to submit a set of data that is not schematized, there are two possibilities: the developer defines a new item type using a custom schema, where this item type holds the proper information, and instances of this type are submitted to the rule engine; and the developer uses XML support to submit the data. As an example of the first option, consider that the developer of an application wants to enable users to create rules around the processing of one of the applications events (non-schematized data). To clarify, a developer wants to craft a type, and provide the type as input to a decision point via a submit. The developer then expects as a response one of the methods on the type to be the action the end-user wants to take. For example, if the object is a soda can, and the methods were Drink, CrushAndDiscard, and SetOnTable, then the conditions would be if"empty" and "recycle bin handy", then CrushAndDiscard. If "not empty", and "not thirsty", then SetOnTable.

[0067] DecisionPoints may optionally require "input scopes". Certain decision points may involve rules constrained to a particular portion of the file system. These DecisionPoints require that any rules attached to them include a relationship to an Input Scope. For instance, the hypothetical "Item Created" decision point above could require an input scope—and would thus be denoting that it requires any rules attached to it to provide a scope within the file system where these item creation inputs would come from.

[0068] The logical building block in the rules-based architecture is the collection. A collection is made up of a "scoping" item and a relationship type. Any items that are the targets of relationships of the specified type emanating from the "scoping" item are considered to be part of the collection rooted at the scoping item.

[0069] RuleSetAttachment items contain methods used for evaluating entire collections simultaneously (e.g., RuleSetAttachment.Evaluate( ) and RuleSetAttachment-.EvaluateIntoCollection( )). These methods the items in the attached input scope as input for the engine to consider. The items are evaluated as if they were submitted individually. The rules engine does not treat collections specially for purposes of evaluation.

[**0070**] Each call to one of the DecisionPoint submit( ) methods or the RuleSetAttachment evaluate( ) methods will cause one or more attached rules to be evaluated by the rule engine. Results from these evaluations are written into a single RuleSetEvaluation item per call.

[**0071**] The RuleSetEvaluation item has a nested element set of RuleResultElements. Each RuleResultElement carries the result name and the structured data necessary to describe the result. These results are application-specific; that is, they are expected to be understandable by the application through/for which the Rule was created and submitted the event. In many cases, the result name carried within the RuleResultElement may actually be a fully-qualified class-level (static) CLR (common language runtime) method name, and the structured data may correspond to parameters for that method. The RuleResultElement type includes an execute( ) method capable of taking this data, and, using CLR reflections, call the named method with the proper parameters. In one implementation, these are limited to setter/getter methods on the properties of the input item as well as static methods found in libraries available within the execution context of the rules customization application.

[**0072**] In another implementation, a new RuleSetEvaluation item is generated for each attachment to the decision point to which the input was submitted.

[**0073**] A FunctionInfo type is shared between the nested types RuleResultElement and Action, the latter being part of the Rule definition.

[**0074**] For those applications wishing to receive results in an XML format, an alternate form of RuleResultElement is provided that can contain the XML data of choice.

[**0075**] Referring now to **FIG. 7**, there is illustrated a system **700** that employs a learning component **702** in accordance with the present invention. The system **700** includeds the customization component **102** that facilitates the exposing of an application generated event in the application **104** to an end-user. The rules component **106** allows the end-user to create one or more rules to associate with the event, which one or more rules facilitate the submission of the application data **108** associated with the event for external processing. Various functions of the application are rule-enabled. The end-user can then create rules for those functions to further manipulate data associated with those functions. Each rule-enabled feature of the application is defined by a decision point, which can be processed externally to return a result to the application that modifies behavior of the application.

[**0076**] The subject invention can employ various artificial intelligence based schemes for carrying out various aspects of the subject invention. For example, a process for determining where to place a decision point can be facilitated via an automatic classifier system and process. A classifier is a function that maps an input attribute vector, x=(x**1**, x**2**, x**3**, x**4**, xn), to a confidence that the input belongs to a class, that is, f(x)=confidence(class). Such classification can employ a probabilistic and/or statistical-based analysis (e.g., factoring into the analysis utilities and costs) to prognose or infer an action that a user desires to be automatically performed.

[**0077**] A support vector machine (SVM) is an example of a classifier that can be employed. The SVM operates by finding a hypersurface in the space of possible inputs, which hypersurface attempts to split the triggering criteria from the non-triggering events. Intuitively, this makes the classification correct for testing data that is near, but not identical to training data. Other directed and undirected model classification approaches include, e.g., naïve Bayes, Bayesian networks, decision trees, and probabilistic classification models providing different patterns of independence can be employed. Classification as used herein also is inclusive of statistical regression that is utilized to develop models of priority.

[**0078**] As will be readily appreciated from the subject specification, the subject invention can employ classifiers that are explicitly trained (e.g., via a generic training data) as well as implicitly trained (e.g., via observing user behavior, receiving extrinsic information). For example, SVM's are configured via a learning or training phase within a classifier constructor and feature selection module. Thus, the classifier(s) can be used to automatically perform a number of functions, including but not limited to determining the location of decision points based in the particular end-user or the application to customized, determining where to place the decision points based on end-user history of decision point placement, and what decision points can be employed based on the type of application. The classifier can be employed to determine what rule to attach to a decision point for a loosely bound model. Similar classifier operations employed for decision points in the loosely bound model can be applied to rules for the tightly bound model.

[**0079**] In a more robust implementation, the classifier performs the complete end-to-end application customization process for the end-user based on end-user preferences and past customizations. Further, the classifier can be used to determine when to use a loosely bound or tightly bound model of application customization and according to a given end-user.

[**0080**] Referring now to **FIG. 8**, there is illustrated a block diagram of a computer operable to execute the disclosed architecture. In order to provide additional context for various aspects of the present invention, **FIG. 8** and the following discussion are intended to provide a brief, general description of a suitable computing environment **800** in which the various aspects of the present invention can be implemented. While the invention has been described above in the general context of computer-executable instructions that may run on one or more computers, those skilled in the art will recognize that the invention also can be implemented in combination with other program modules and/or as a combination of hardware and software.

[**0081**] Generally, program modules include routines, programs, components, data structures, etc., that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods can be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, as well as personal computers, hand-held computing devices, microprocessor-based or programmable consumer electronics, and the like, each of which can be operatively coupled to one or more associated devices.

[**0082**] The illustrated aspects of the invention may also be practiced in distributed computing environments where certain tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

[0083] A computer typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media can comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital video disk (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer.

[0084] Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

[0085] With reference again to **FIG. 8**, there is illustrated an exemplary environment **800** for implementing various aspects of the invention that includes a computer **802**, the computer **802** including a processing unit **804**, a system memory **806** and a system bus **808**. The system bus **808** couples system components including, but not limited to, the system memory **806** to the processing unit **804**. The processing unit **804** can be any of various commercially available processors. Dual microprocessors and other multi-processor architectures may also be employed as the processing unit **804**.

[0086] The system bus **808** can be any of several types of bus structure that may further interconnect to a memory bus (with or without a memory controller), a peripheral bus, and a local bus using any of a variety of commercially available bus architectures. The system memory **806** includes read only memory (ROM) **810** and random access memory (RAM) **812**. A basic input/output system (BIOS) is stored in a non-volatile memory **810** such as ROM, EPROM, EEPROM, which BIOS contains the basic routines that help to transfer information between elements within the computer **802**, such as during start-up. The RAM **812** can also include a high-speed RAM such as static RAM for caching data.

[0087] The computer **802** further includes an internal hard disk drive (HDD) **814** (e.g., EIDE, SATA), which internal hard disk drive **814** may also be configured for external use in a suitable chassis (not shown), a magnetic floppy disk drive (FDD) **816**, (e.g., to read from or write to a removable diskette **818**) and an optical disk drive **820**, (e.g., reading a CD-ROM disk **822** or, to read from or write to other high capacity optical media such as the DVD). The hard disk drive **814**, magnetic disk drive **816** and optical disk drive **820** can be connected to the system bus **808** by a hard disk drive interface **824**, a magnetic disk drive interface **826** and an optical drive interface **828**, respectively. The interface **824** for external drive implementations includes at least one or both of Universal Serial Bus (USB) and IEEE 1394 interface technologies.

[0088] The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions, and so forth. For the computer **802**, the drives and media accommodate the storage of any data in a suitable digital format. Although the description of computer-readable media above refers to a HDD, a removable magnetic diskette, and a removable optical media such as a CD or DVD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as zip drives, magnetic cassettes, flash memory cards, cartridges, and the like, may also be used in the exemplary operating environment, and further, that any such media may contain computer-executable instructions for performing the methods of the present invention.

[0089] A number of program modules can be stored in the drives and RAM **812**, including an operating system **830**, one or more application programs **832**, other program modules **834** and program data **836**. All or portions of the operating system, applications, modules, and/or data can also be cached in the RAM **812**.

[0090] It is appreciated that the present invention can be implemented with various commercially available operating systems or combinations of operating systems.

[0091] A user can enter commands and information into the computer **802** through one or more wired/wireless input devices, e.g., a keyboard **838** and a pointing device, such as a mouse **840**. Other input devices (not shown) may include a microphone, an IR remote control, a joystick, a game pad, a stylus pen, touch screen, or the like. These and other input devices are often connected to the processing unit **804** through an input device interface **842** that is coupled to the system bus **808**, but can be connected by other interfaces, such as a parallel port, an IEEE 1394 serial port, a game port, a USB port, an IR interface, etc.

[0092] A monitor **844** or other type of display device is also connected to the system bus **808** via an interface, such as a video adapter **846**. In addition to the monitor **844**, a computer typically includes other peripheral output devices (not shown), such as speakers, printers etc.

[0093] The computer **802** may operate in a networked environment using logical connections via wired and/or wireless communications to one or more remote computers, such as a remote computer(s) **848**. The remote computer(s) **848** can be a workstation, a server computer, a router, a personal computer, portable computer, microprocessor-based entertainment appliance, a peer device or other common network node, and typically includes many or all of the elements described relative to the computer **802**, although, for purposes of brevity, only a memory storage device **850** is illustrated. The logical connections depicted include wired/wireless connectivity to a local area network (LAN) **852** and/or larger networks, e.g., a wide area network (WAN) **854**. Such LAN and WAN networking environments are commonplace in offices, and companies, and facilitate enterprise-wide computer networks, such as intranets, all of which may connect to a global communication network, e.g., the Internet.

[0094] When used in a LAN networking environment, the computer **802** is connected to the local network **852** through a wired and/or wireless communication network interface or adapter **856**. The adaptor **856** may facilitate wired or wireless communication to the LAN **852**, which may also include a wireless access point disposed thereon for communicating with the wireless adaptor **856**. When used in a WAN networking environment, the computer **802** can include a modem **858**, or is connected to a communications server on the LAN, or has other means for establishing communications over the WAN **854**, such as by way of the Internet. The modem **858**, which can be internal or external and a wired or wireless device, is connected to the system bus **808** via the serial port interface **842**. In a networked environment, program modules depicted relative to the computer **802**, or portions thereof, can be stored in the remote memory/storage device **850**. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers can be used.

[**0095**] The computer **802** is operable to communicate with any wireless devices or entities operatively disposed in wireless communication, e.g., a printer, scanner, desktop and/or portable computer, portable data assistant, communications satellite, any piece of equipment or location associated with a wirelessly detectable tag (e.g., a kiosk, news stand, restroom), and telephone. This includes at least Wi-Fi and Bluetooth™ wireless technologies. Thus, the communication can be a predefined structure as with conventional network or simply an ad hoc communication between at least two devices.

[**0096**] Wi-Fi, or Wireless Fidelity, allows connection to the Internet from a couch at home, a bed in a hotel room or a conference room at work, without wires. Wi-Fi is a wireless technology like a cell phone that enables such devices, e.g., computers, to send and receive data indoors and out; anywhere within the range of a base station. Wi-Fi networks use radio technologies called IEEE 802.11 (a, b, g, etc.) to provide secure, reliable, fast wireless connectivity. A Wi-Fi network can be used to connect computers to each other, to the Internet, and to wired networks (which use IEEE 802.3 or Ethernet). Wi-Fi networks operate in the unlicensed 2.4 and 5 GHz radio bands, with an 11 Mbps (802.11a) or 54 Mbps (802.11b) data rate or with products that contain both bands (dual band), so the networks can provide real-world performance similar to the basic 10BaseT wired Ethernet networks used in many offices.

[**0097**] Referring now to **FIG. 9**, there is illustrated a schematic block diagram of an exemplary computing environment **900** in accordance with the present invention. The system **900** includes one or more client(s) **902**. The client(s) **902** can be hardware and/or software (e.g., threads, processes, computing devices). The client(s) **902** can house cookie(s) and/or associated contextual information by employing the present invention, for example. The system **900** also includes one or more server(s) **904**. The server(s) **904** can also be hardware and/or software (e.g., threads, processes, computing devices). The servers **904** can house threads to perform transformations by employing the present invention, for example. One possible communication between a client **902** and a server **904** can be in the form of a data packet adapted to be transmitted between two or more computer processes. The data packet may include a cookie and/or associated contextual information, for example. The system **900** includes a communication framework **906** (e.g., a global communication network such as the Internet) that can be employed to facilitate communications between the client(s) **902** and the server(s) **904**.

[**0098**] Communications can be facilitated via a wired (including optical fiber) and/or wireless technology. The client(s) **902** are operatively connected to one or more client data store(s) **908** that can be employed to store information local to the client(s) **902** (e.g., cookie(s) and/or associated contextual information). Similarly, the server(s) **904** are operatively connected to one or more server data store(s) **910** that can be employed to store information local to the servers **904**.

[**0099**] What has been described above includes examples of the present invention. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the present invention, but one of ordinary skill in the art may recognize that many further combinations and permutations of the present invention are possible. Accordingly, the present invention is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a transitional word in a claim.

What is claimed is:

1. A system that facilitates data management of an application, comprising:

a customization component that facilitates exposing of an application-generated event to an end-user; and

a rules component that allows the end-user to create one or more rules to process the event, which the one or more rules facilitate the submission of application data associated with the event for processing.

2. The system of claim 1, the event is associated with a rules-based decision point.

3. The system of claim 1, the application data is processed at least one of externally and internally to return a result to the application that modifies behavior of the application.

4. The system of claim 1, the end-user creates explicit rules that define how the event will be processed and the application customized.

5. The system of claim 1, further comprising a learning component that facilitates the application learning end-user behavior, and captures the behavior in the form of rules.

6. The system of claim 1, the end-user can explicitly modify the one or more rules.

7. The system of claim 1, the application exposes an item that is a decision point at which the end-user can customize behavior of the application.

8. The system of claim 7, the application exposes a name that identifies context of the decision in which the one or more rules are applied.

9. The system of claim 7, the application exposes a signature that describes input data types and an expected result type.

10. The system of claim 7, the decision point is registered when the application is installed.

11. The system of claim 1, the customization component facilitates selection of one of a loosely-bound model and a tightly-bound model for customization of the application.

12. The system of claim 1, wherein an application that is tightly bound chooses a subset of the one or more rules created by the end-user, and directly executes the subset to produce a customization decision.

13. The system of claim 1, wherein the application exposes a decision point that provides a name to identify the nature of a decision that is to be made, an application name that identifies the owning application, and a rule constraint that identifies the kind of rules allowed for the decision point.

14. A computer readable medium having stored thereon computer executable instructions for carrying out the system of claim 1.

15. A computer readable medium having stored thereon the components of claim 1.

16. A computer that employs the system of claim 1.

17. A system that facilitates data management of an application, comprising:

a customization component that facilitates customization of the application by an end-user by exposing of an application-generated decision point to the end-user; and

a rules component that facilitates the creation of one or more rules by the end-user to associate with the decision point, which one or more rules facilitate the submission of application data associated with the decision point for external processing.

18. The system of claim 17, the end-user can explicitly modify the one or more rules.

19. The system of claim 17, the application exposes a name that identifies context of the decision point in which the one or more rules are applied and a signature that describes input data types and an expected result type.

20. The system of claim 17, wherein the application directly executes the one or more rules to produce a customization decision.

21. The system of claim 17, wherein the application identifies one or more correct rulesets in accordance with a query for the one or more correct rulesets that correspond to the application.

22. The system of claim 21, wherein the query corresponds to the application, user, decision, and input that is to be provided to the one or more rules.

23. The system of claim 21, wherein the decision point is associated with a decision point input method that is invoked by a calling application.

24. The system of claim 17, the rules component facilitates associating a rule constraint with the decision point, which rule constraint filters which rules will run for a given invocation and a particular input.

25. A computer-readable medium having computer-executable instructions for performing a method of facilitating end-user customization of an application, the method comprising:

exposing an application-generated decision point of the application;

creating one or more rules that operate on data of the decision point;

associating the one or more rules with the decision point; and

processing the one or more rules to expose the data for external processing.

26. The method of claim 25, further comprising directly executing the one or more rules to produce a customization decision.

27. The method of claim 25, further comprising querying for one or more correct rules that correspond to the application.

28. The method of claim 25, wherein the end-user explicitly modifies the one or more rules.

29. The method of claim 25, further comprising invoking a decision point input method that is called by a calling application.

30. The method of claim 25, further comprising filtering which rules will run for a given invocation by processing a rule constraint associated with the decision point.

31. The method of claim 25, further comprising exposing a name that identifies context of the decision point in which the one or more rules are applied and a signature that describes input data types and an expected result type.

32. The method of claim 25, further comprising associating a rule constraint with the decision point and one or more of the rules.

33. The method of claim 25, further comprising the acts of,

directly invoking an execute method on the one or more rules;

passing an input item that is an input to the one or more rules; and

passing a rule constraint that enforces an input and an output.

34. A method of facilitating end-user customization of an application, comprising:

exposing an application-generated decision point of the application;

creating by an end-user one or more rules that operate on data of the application;

in first mode, attaching the one or more rules to a decision point and calling a method on an item of the decision point to expose the data for processing; and

in a second mode, identifying correct one or more rules for the application, and processing the correct one or more rules directly to expose the data for processing.

35. The method of claim 34, further comprising filtering which of the one or more rules will run for a given invocation by processing a rule constraint associated with the decision point.

36. The method of claim 34, further comprising employing both the first mode and the second mode for end-user customization of the application.

* * * * *