SYSTEM AND METHOD FOR DEVELOPING BUSINESS PROCESS POLICIES

Cross Reference to Related Applications

This application claims priority to U.S. provisional application entitled "System and Method for Developing Business Process Policies," Serial No. 60/303,054, filed July 5, 2001, which is incorporated herein by reference.

Technical Field

The methods, systems, and computer readable media described herein relate generally to computer programming and more particularly to defining and managing objects in business process policy computing.

Background of the Invention

Dynamic real time management of business information facilitates improving business performance and process management solutions. Various activities are associated with handling business events and situations. For example, a user may be alerted when a business event occurs, data associated with the business event may be presented using a visual tool (e.g., graphical user interface (GUI)), automated processing may diagnose problems associated with or reported by the business event, and automated processing may attempt to resolve issues associated with the business event.

One business event can have different meanings, reporting requirements, processing requirements, and so on to different people, at different times, from different points of view. Conventionally, it was difficult, if possible at all, to perform such point of view selective, customizable, business process policy logic. Thus, systems and methods that facilitate developing business process policies to provide very granular support that facilitates applying intelligence and solutions according to different factors are still required.

Summary of the Invention

The following presents a simplified summary of methods, systems, and computer readable media associated with business process policy objects. This summary is not an

extensive overview and is not intended to identify key or critical elements of the methods, systems, and/or media or to delineate the scope of the methods, systems, and media. It conceptually identifies the methods, systems, and media in a simplified form as a prelude to the more detailed description that is presented later.

This application concerns methods, systems, application programming interfaces (APIs), GUIs, and computer readable media for defining and managing objects that model and implement business process policy logic. Employing objects in business process policy computing facilitates managing and optimizing business processes and performance in a user and/or role customized manner.

This application also concerns a development methodology for abstracting user and role level details out of business process policies and into policy definition objects. Furthermore, the development methodology concerns providing individual policy definition object values in user policy objects. The methodology focuses on abstracting user details and abstracting role details out of the execution logic in a business process policy object to facilitate making business process policy objects more general while at the same time facilitating user, role, and context based processing for business process policies. When employing the methodology described herein, a business process policy object should have a minimal focus on the identity of a user, the role of a user, how information handling should be customized for that user and/or role, and how the information should be presented based on the user identity and/or role. Instead, the business process policy should focus on making information management more flexible and making data management more flexible. To facilitate business process policy logic being managed by an external set of computer components (e.g., policy managers, schedulers), the user and role level details are abstracted out of the business process policy. Thus, the policy managers, schedulers, and so on, can manage different user configuration choices, including but not limited to, desired visualizations, desired notifications, and desired default handling of business events.

Dynamic, real time management of business information facilitates improving business performance and process management solutions. Processing business events in various situations requires very granular support to facilitate applying intelligence and solutions according to different relevant factors. These factors can include, but are not limited to, the identity of the user, the role of the user, the financial status of the user, the relationship of a user to other users, and so on. These factors, which may be referred to as various "points of view" affect processing associated with alerting a user, visualizing an event

or situation, diagnosing problems associated with an event and/or situation and resolving problems associated with the event and/or situation. Thus, these factors affect what information is presented in response to an event, how to present that information, who to present the information to, and what intelligence solution to apply to the problem.

Isolating business logic into a business process policy, and associating a business process policy with a policy definition object and one or more user policy objects facilitates performing the business process policy logic in both a generic and a point of view selective, individualized manner. Thus, this application describes systems and methods that help users specify what business events are important to them, when those business events are important, and in what way those events are important. The systems and methods described herein facilitate efficiently, and at a granular level, simplifying managing information that is received, information that is generated, information that is inferred, and/or information that is predicted by business process policies acting on business events.

In one example, the systems and methods described herein facilitate defining business process policies. This business process policy definition facilitates integrating information from a number of sources. The definition also facilitates removing individual point of view requirements and personalizations from business process policy logic by defining a separate object type that defines the points of individualization applicable to a business process policy. Furthermore, another class of objects is defined, where this class facilitates implementing instances of the points of individuality for a specific user.

Thus, one aspect of this application concerns a system for abstracting point of view specific logic out of a business process policy object. The system includes a business process policy object that implements a business intelligence for handling a business event from a generic point of view. The system also includes a policy definition object related to the business process policy object. The policy definition object implements a business intelligence for handling the business event from a specific point of view. The system also includes a user policy object associated with the policy definition object. The user policy object stores a data that facilitates executing the business intelligence in the policy definition object.

Another aspect of the application concerns a computer implemented method for separating point of view specific logic from generic view logic. The method includes creating a business process policy logic for handling business events. The method also includes abstracting out of the business process policy logic a point of view specific logic and

a generic view logic. Once the two logics are abstracted, the method includes implementing the generic view logic in a business process policy object, implementing the point of view specific logic in a policy definition object, and storing point of view specific data in a user policy object to facilitate executing the point of view specific logic.

Certain illustrative aspects of the methods, systems, APIs, GUIs, and computer readable media 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 methods, systems, APIs, GUIs, and media may be employed and thus the examples are intended to include such aspects and equivalents. Other advantages and novel features may become apparent from the following detailed description when considered in conjunction with the drawings.

Brief Description of the Drawings

Figure 1 illustrates an example computing environment with which example abstracting systems and methods can interact.

Figure 2 illustrates three objects interacting to facilitate isolating business process policy logic from specific points of views.

Figure 3 illustrates an example system in which business process policy objects and policy definition objects operate.

Figure 4 illustrates an example system in which business process policy objects, policy definition objects, and user policy objects interact.

Figure 5 illustrates an example set of objects interacting with a key performance indicator (KPI).

Figure 6 illustrates an example set of objects facilitating event handling.

Figure 7 is a flow chart of an example method for implementing business process policy logic in an object.

Figure 8 is a flow chart of a portion of an example method for performing business process policy logic.

Figure 9 illustrates an example GUI employed in object-oriented business event handling.

Figure 10 illustrates an example API employed in object-oriented business event handling.

Detailed Description

Example methods, systems, APIs, GUIs, and computer readable media are now described with reference to the drawings, where 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 facilitate thoroughly understanding the methods, systems, and so on. It may be evident, however, that the methods, systems, and so on can be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to simplify description.

Figure 1 illustrates a computer 100 that includes a processor 102, a memory 104, a disk 106, input/output ports 110, and a network interface 112 operably connected by a bus 108. Executable components of the systems described herein may be located on a computer like computer 100. Similarly, computer executable methods described herein may be performed on a computer like computer 100. Furthermore, business process policy objects, policy definition objects, and user policy objects may reside on a computer like computer 100 and/or be processed by a computer like computer 100. It is to be appreciated that other computers may also be employed with the systems and methods described herein.

The processor 102 can be a variety of various processors including dual microprocessor and other multi-processor architectures. The memory 104 can include volatile memory and/or non-volatile memory. The non-volatile memory can include, but is not limited to, read only memory (ROM), programmable read only memory (PROM), electrically programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), and the like. Volatile memory can include, for example, random access memory (RAM), synchronous RAM (SRAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR SDRAM), and direct RAM bus RAM (DRRAM). The disk 106 can include, but is not limited to, devices like a magnetic disk drive, a floppy disk drive, a tape drive, a Zip drive, a flash memory card, and/or a memory stick. Furthermore, the disk 106 can include optical drives like a compact disk ROM (CD-ROM), a CD recordable drive (CD-R drive), a CD rewriteable drive (CD-RW drive) and/or a digital versatile ROM drive (DVD ROM). The memory 104 can store processes 114 and/or data 116, for example. The disk 106 and/or memory 104 can store an operating system that controls and allocates resources of the computer 100.

The bus 108 can be a single internal bus interconnect architecture and/or other bus architectures. The bus 108 can be of a variety of types including, but not limited to, a

memory bus or memory controller, a peripheral bus or external bus, and/or a local bus. The local bus can be of varieties including, but not limited to, an industrial standard architecture (ISA) bus, a microchannel architecture (MSA) bus, an extended ISA (EISA) bus, a peripheral component interconnect (PCI) bus,. a universal serial (USB) bus, and a small computer systems interface (SCSI) bus.

The computer 100 interacts with input/output devices 118 via input/output ports 110. The input/output devices 118 can include, but are not limited to, a keyboard, a microphone, a pointing and selection device, cameras, video cards, displays, and the like. The input/output ports 110 can include but are not limited to, serial ports, parallel ports, and USB ports.

The computer 100 can operate in a network environment and thus is connected to a network 120 by a network interface 112. Through the network 120, the computer 100 may be logically connected to a remote computer 122. The network 120 includes, but is not limited to, local area networks (LAN), wide area networks (WAN), and other networks. The network interface 112 can connect to local area network technologies including, but not limited to, fiber distributed data interface (FDDI), copper distributed data interface (CDDI), ethernet/IEEE 802.3, token ring/IEEE 802.5, wireless/IEEE 802.11 and the like. Similarly, the network interface 112 can connect to wide area network technologies including, but not limited to, point to point links, and circuit switching networks like integrated services digital networks (ISDN), packet switching networks, and digital subscriber lines (DSL).

Turning now to Figure 2, a collection 200 of objects interacting to facilitate isolating business process policy logic from specific points of view is illustrated. A business process policy object 210 is illustrated communicating with a policy definition object 220 and a user policy object 230.

A business process policy object 210 models and implements intelligence (e.g., decision making) applied in understanding, analyzing, and/or responding to business events. A business event can be, for example, a message between an information source and an intended target (e.g., workflow manager, enterprise monitor) that describes a significant business activity. Business events can also be modeled in objects, and can be communicated to business process policy objects via, for example, object oriented messaging. Business events can be associated with business logic processing and can be involved, for example, in reporting a state, in noting a status change, in providing data, and so on.

By way of illustration, a business process policy object 210 can facilitate automatically performing actions like responding to an inventory change, predicting sales

activity, generating appropriate discounts, and scheduling deliveries. These and similar actions can be triggered by business events and/or messages, for example. Various entities may desire various processing to be performed in association with an event based on their point of view. The types of business events for which a business process policy object 210 can implement logic include, but are not limited to, reference events, change events, threshold events, task completion events, and task failure events. Furthermore, a business process policy object 210 can model and implement logic that deals with individual events and/or aggregations of events. The systems and methods described herein facilitate abstracting point of view specific logic and/or data out of a business process policy and/or a business process policy object 210. Thus, the business process policy and/or business process policy object 210 can be isolated from specific points of view and be more generally useable. Furthermore, other objects (e.g., policy definition objects (PDO) 220, user policy objects (PUO) 230) can hold the point of view specific logic and/or data to facilitate localizing point of view specific processing.

An example schema of a business event facilitates identifying the description of and details concerning the business event being reported. The event may be related to object happenings, business processes, and/or intelligence (e.g., facts) gathered, for example. One example XML schema for a business event processed by a business process policy object 210 is:

```
< COMPANY A>
    <EVENT NAME> . . . </EVENT NAME>
    <NAME SPACE> . . . </NAME SPACE>
    <OBJECT NAME>
        <CLASS NAME> . . . </CLASS NAME>
            <OBJECT ID>
                <NAME_VALUE>
                    <NAME> . . . </NAME>
                    <TYPE> . . . </TYPE>
                    <VALUE> . . . </VALUE>
                </NAME_VALUE>
                <NAME_VALUE>
                    <NAME> . . . </NAME>
                    <TYPE> . . . </TYPE>
```

```
                    <VALUE> . . . </VALUE>
                </NAME_VALUE>
            </OBJECT ID>
        </OBJECT NAME>
        <MSGTEXT> . . . </MSGTEXT>
            <PROPERTY>
                <NAME> . . . </NAME>
                <TYPE> . . . </TYPE>
                <OLD VALUE> . . . </OLD VALUE>
                <NEW VALUE> . . . </NEW VALUE>
            </PROPERTY>
    </COMPANY A>
```

Business logic can be selectively performed. Business logic processing can include, but is not limited to, invoking one or more methods of a business process policy object 210, accessing one or more data items of a business process policy object 210, and/or updating one or more of the data items. Which methods to invoke and/or which data items to access, update, and/or display can depend, for example, on the point of view of an entity. Business logic that is modeled and implemented in a business process policy object 210 can be expressed, for example, in IF/THEN rules, rule capturing tables, and neural networks. By way of illustration, simple inventory logic may be modeled and implemented by the following IF/THEN rule.

```
IF inventory_item < X THEN
    order more inventory_item
END IF
```

Similarly, simple discount logic may be modeled and implemented by the following IF/THEN rule.

```
IF order value > relevant limit THEN
    generate 10% discount
ELSE
    generate standard discount
END IF
```

Business logic may be associated with a business indicator. In some cases, the business indicator may even be a key performance indicator. These indicators may have different importance to different people. For example, one business indicator (e.g., day to day sales) is a singular, discrete, unique value. But it can have different relevance and importance to different people according to, for example, their position in the company, the point in time, and other information that person has. A change in day to day sales may mean one thing to a CEO, another thing to a person responsible for cash flow, and yet another thing to inventory and shipping personnel depending on their "point of view" (e.g., role, context, knowledge, needs, relationships). Thus, while general business process policy logic associated with current day to day sales may reside in a business process policy object 210, individual actions (e.g., reports, updates, processes) may be defined in a policy definition object 220 and may have certain aspects implemented in a user policy object 230.

A business process policy can handle a business event. A business process policy might raise an alert at a first level for a first role (e.g., stock boy), at a second level for a second role (e.g., inventory manager), and at a third level for a third role (e.g., manufacturing capacity planner). Thresholds for the different roles will not be stored in the business process policy, but rather will be stored in a policy definition object 220 and/or a user policy object 230. Thus, the business process policy can be executed for relevant, registered users, with generic processing occurring in a point of view free manner, and with point of view specific processing occurring in a point of view specific manner in conjunction with logic and/or data implemented in a policy definition object 220 or user policy object 230.

Policy definition objects 220 specify how policies should be written to expose dynamic threshold control and to permit sharing of business process policy intelligence encapsulated in the form of sharable objects. Policy definition objects 220 specify a methodology and architecture for how business process policies should be defined in the form of objects. The objects include encapsulating data employed by business process policies at execution time, sharable data, and interfaces to permit modification of the logic and/or data by computer components like graphical user interfaces, application programming interfaces, and command line interfaces. Policy definition objects 220 encapsulate information that allows a policy dispatcher to execute a business process policy in the context of different users. This facilitates systems and methods described herein managing event information to apply different policies depending on factors like user and/or role based paradigms.

Policy user objects 230 specify a methodology and architecture for how user specific data for a policy should be defined in the form of objects that include thresholds and data. The thresholds can be, for example, a set of exposed and modifiable parameters that facilitate changing business process policy behavior depending on external factors like events, applications, user context, and other business process policies. The data facilitates encapsulating point of view specific (e.g., user, role, relationship) data needed by a business process policy at execution time. The thresholds and/or data can be modified by entities, including but not limited to, graphical user interfaces, application programming interfaces, and command line interfaces.

A policy definition object 220 facilitates customizing business process policy intelligence by externalizing threshold values on a user basis and/or role basis. The user policy object 230 holds data that facilitates the policy definition object 220 customizing the business process policy intelligence in this manner. The policy definition object 220 encapsulates data needed by a business process policy when a business process policy object 210 is executed. Thus, the policy definition object 220 and the user policy object 230 separate point of view specific data and/or logic from the generic business process policy 210 logic. A policy definition object 220 and/or user policy object 230 can be defined dynamically at run time and thus can be generated at run time, even by business process policies 210 as needed.

Thus, the collection 200 can be employed in a system for abstracting point of view specific logic out of a business process policy object 210. The system includes a business process policy object 210 that implements business intelligence (e.g., logic, if/then rules, neural networks, tables) for handling a business event (e.g., change event, reference event, task completion event, task failure event, threshold event) from a generic point of view. The system also includes a policy definition object 220 that can be related to the business process policy object 210. The policy definition object 220 implements a second business intelligence that handles the business event from a specific point of view. This specific point of view may depend, for example, on the role, context, knowledge, needs, and/or relationships of the entities associated with the policy definition object 220. The system also includes a user policy object 230 associated with the policy definition object 220. The user policy object 230 stores data that facilitates the policy definition object 220 executing a second business logic in a point of view specific manner.

The generic processing that can be performed by the business process policy object 210 includes, but is not limited to, visualizing an event, reporting out an event, correlating events, processing events, and predicting events. Similarly, the point of view specific business intelligence performed by the policy definition object 220 includes, but is not limited to, visualizing an event, reporting out an event, correlating events, processing events, and predicting events.

In one example, data stored in the user policy object 230 includes, but is not limited to, a threshold value, key performance indicator identifiers, business process policy object 210 identifiers, policy definition objects 220 identifiers, point of view specific policy values, and context specific policy values. It is to be appreciated that the data stored in the user policy object 230 can be shared. For example, the data may be shared between various roles and/or users. Thus, although the business process policy 210 stores generic logic, the combination of a policy definition object 220 and a user policy object 230 can store individual points of view logic and/or data and/or aggregations of logic and/or data.

The data stored in the user policy object 230 can be modified by, for example, a user interacting with a graphical user interface, a user and/or process interacting with a command line interface, a user and/or process interacting with an application programming interface, and/or a user, object, and/or process interacting with an object oriented message. The data stored in the user policy object 230 and/or the logic implemented in the policy definition object 220 can be specific to one or more points of view. These points of view can include, but are not limited to, a user specific point of view, a role specific point of view, a context specific point of view, a time related point of view, a relationship based point of view.

Referring now to Figure 3, an example system 300 that employs business process policy objects (e.g., 330, 332, 334) and policy definition objects (e.g., 322, 324) to facilitate general and point of view specific business event handling is illustrated. A business event can be, for example, a message between an information source and work flow manager that describes a significant business activity. Business process policies are invoked to process business events. Business process policies can benefit from abstraction and customization. Thus, system 300 facilitates removing customized processing and/or data from a business process policy and storing it in a policy definition object and/or user policy object. Thus, a business process policy can be related to policy definition objects and executed in one or more user contexts associated with one or more user policy objects.

The event manager 340 can receive events including, but not limited to, reference events, change events, threshold events, task completion events, and task failure events. A reference event, which may be a discrete event, can provide information like a date when a company files a financial disclosure, or a notification that a company has filed its financial disclosure. A change event can relate prior intelligence to other intelligence that has not yet been related to other events. For example, a change event may provide information concerning when a product price page changes, or when a company stock price changes. A threshold event facilitates simple levels of correlation between current knowledge and prior knowledge. For example, a threshold event may provide information concerning when a company's stock price has gone up or down 10% over the previous price. A task completion event relates to business process intelligence and thus may provide information concerning when an on-going task has completed (e.g., informing a business process policy object that a financial disclosure data download has completed). Each of various entities may desire processing to be performed based on their unique needs. For example, entities may have unique needs in areas including, but not limited to, data capture, data display, data aggregation, data integration, data prediction, and so on.

Events can be programmatically related and aggregated in a business process policy object to facilitate comprehensive command and control by applying business intelligence. While there are generic aspects of business intelligence, there are also individualized aspects of business intelligence. These individualized aspects facilitate applying general business logic in user, role, identity and context specific manners without compromising the generality of business process policy in which the intelligence is located.

The system 300 includes a business process policy manager 310 that facilitates dynamically defining and instantiating a policy definition object. The business process policy manager 310 also facilitates managing the life cycle of a policy definition object by producing life cycle management requests including, but not limited to, creating a policy definition object, deleting a policy definition object, updating a policy definition object, and querying a policy definition object. When a policy definition object is queried, the value(s) for which the query was made are returned to the business process policy manager 310 and/or other designated destinations.

The system 300 also includes a business process policy engine 320 that interacts with the business process policy 310. Thus, the business process policy engine 320 manages policy definition objects and exposes policy definition objects. In one example, the business

process policy engine 320 exposes policy definition objects within various common environments. Thus, one or more policy definition objects (e.g., 322, 324) are illustrated residing within the business process policy engine 320. While blocks 322 and 324 are illustrated inside the business process policy engine 320, it is to be appreciated that the policy definition objects can be associated with the business process policy engine 320 without necessarily residing inside the business process policy engine 320. The business process policy engine 320 also interacts with one or more business process policy objects (e.g., 330, 332, 334). The business process policy objects handle events that are filtered through the business process policy dispatcher 350 and the event manager 340. The dispatcher 350 matches an event policy identifier with registered policy definition objects. Using policy definition object execution information, the dispatcher 350 triggers an appropriate business process policy. The business process policy is then executed in one or more point of view specific manners (e.g., user context, user role) with various values as defined in a policy definition object and implemented in a user policy.

It is to be appreciated that computer executable components of system 300 and/or collection 200 can be stored on a computer readable medium.

Turning now to Figure 4, a system 400 in which business process policy objects, policy definition objects, and user policy objects interact is illustrated. The system 400 includes a business process policy manager 410 that facilitates managing the life cycle of policy definition objects by creating life cycle management requests including, but not limited to, create, update, query, and delete requests. These requests are forwarded from the business process policy manager 410 to a request manager 430 in a policy manager engine 420. When a create request is received, a policy definition object (e.g., PDO 440) can be dynamically created. The policy definition object may also be created dynamically through the operation of a business process policy 450. When the policy definition object 440 is created, a user policy object 460 is also created to facilitate storing point of view specific data to facilitate executing point of view specific logic in the policy definition object.

When a delete request is received by the request manager, a policy definition object (e.g., PDO 442) is deleted and user policy objects related to the policy definition object are also deleted.

A user policy object class 460 is created to facilitate relating one or more user policy objects to policy definition objects. Instances of the user policy object class 460 (e.g., user policy object 470, user policy object 472) are instantiated to hold policy and threshold values

for a specific entity (e.g., user, context, point of view). Thus, in Figure 4, a user policy object instance 470 is illustrated. This instance 470 is an instance of the class 460 and holds user A policy and threshold values. Similarly, a user policy object 472 has been instantiated from the class 460 and holds policy and threshold value information for a user B. In this way, a business process policy can have user specific information abstracted out of the business process policy through the interaction of a policy definition object and a user policy object.

Figure 5 illustrates a collection 500 of objects. The collection 500 includes a business process policy object 510 interacting with a policy definition object 520 and a user policy object 530. The policy definition object 520 contains an identifier of a key performance indicator 540 (KPI) associated with the business process policy 510. The KPI 540 status can be changed by the business process policy 510 on an as needed basis.

Figure 6 illustrates an example set of objects facilitating event handling. An event 610 arrives at an event distributor 620. The event distributor 620 then enlists the support of a business process policy and/or business process policy object 630, a user policy object 640 and/or policy definition object 650. The three objects then cooperate so that point of view specific processing can be extracted out of the business process policy 630. The business process policy 630 can thus have a more generic view and be more generally usable.

In view of the exemplary systems shown and described herein, methodologies that are implemented will be better appreciated with reference to the flow diagrams of Figures 7 and 8. While for purposes of simplicity of explanation, the illustrated methodologies are shown and described as a series of blocks, it is to be appreciated that the methodologies are not limited by the order of the blocks, as some blocks can occur in different orders and/or concurrently with other blocks from that shown and described. Moreover, less than all the illustrated blocks may be required to implement an example methodology. Furthermore, additional and/or alternative methodologies can employ additional, not illustrated blocks.

In the flow diagrams, rectangular blocks denote "processing blocks" that may be implemented, for example, in software. Similarly, the diamond shaped blocks denote "decision blocks" or "flow control blocks" that may also be implemented, for example, in software. Alternatively, and/or additionally, the processing and decision blocks can be implemented in functionally equivalent circuits like a digital signal processor (DSP), an application specific integrated circuit (ASIC), and the like.

A flow diagram does not depict syntax for any particular programming language, methodology, or style (e.g., procedural, object-oriented). Rather, a flow diagram illustrates

functional information one skilled in the art may employ to program software, design circuits, and so on. It is to be appreciated that in some examples, program elements like temporary variables, routine loops, and so on are not shown.

In one example, methodologies can be implemented as computer executable instructions and/or operations and the instructions and/or operations can be stored on computer readable media including, but not limited to, an application specific integrated circuit (ASIC), a compact disc (CD), a digital versatile disk (DVD), a random access memory (RAM), a read only memory (ROM), a programmable read only memory (PROM), an electronically erasable programmable read only memory (EEPROM), a disk, a carrier wave, and a memory stick.

Referring now to Figure 7, an example method 700 for implementing business process policy logic in an object is illustrated. The method 700 includes, at 710, creating a business process policy logic. This business process policy logic is responsible for handling one or more business events. At 720, a point of view specific logic and a generic view logic are abstracted from the business process policy logic. At 730, the generic logic is implemented, for example, in a business process policy and/or a business process policy object. Similarly, at 740, the point of view specific logic is implemented in a policy definition object that can be related by, for example, identifiers and/or relationships with the business process policy object in which the generic view logic was implemented. At 750, point of view specific data (e.g., thresholds, user data, role data) is stored in a user policy object. Storing this data in the user policy object facilitates executing the point of view specific logic implemented in, for example, the policy definition object.

To facilitate having the business process policy and/or business process policy logic, the policy definition object, and the user policy object work together, in one example, the policy definition object may be registered with a policy dispatcher. Registering a policy definition object exposes one or more of the generic view logic, the point of view specific logic, and/or the view specific data. In one example, these items are exposed to various common environments.

Figure 8 illustrates a portion of an example method 800 for performing business process policy logic. At 860, an object is registered with the dispatcher to expose the logic to, for example, a common environment. Registering the object with the dispatcher can also facilitate a collection of objects (e.g., BPP, PDO, PUO) working together to facilitate abstracting generic logic and point of view specific logic out of a business process policy. At

870, the method receives a business event. The event can be, for example, a reference event, a change event, a threshold event, a task completion event, a task failure event, and so on.

At 880, the event is mapped to a policy definition object. At 890, a business process policy associated with the event and/or the policy definition object is executed from one or more points of view. Thus, the business process policy may be executed in one or more contexts. These points of view and/or contexts can be supported through data stored in a user policy object and/or through logic implemented in policy definition objects to which the business event was mapped. At 895, a determination is made concerning whether there is another event to process. If the determination at 895 is no, then processing can conclude. But if the determination at 895 is yes, then processing can return to 870.

In one example, the life cycle of a policy definition object registered at 860 can be managed through one or more life cycle management requests. These life cycle management requests can include, but are not limited to, create requests, delete requests, update requests, and query requests. It is to be appreciated that computer executable aspects of the method 700 and/or the method 800 can be stored in computer executable instructions on a computer readable medium.

Figure 9 illustrates an example GUI 900 that facilitates accessing, for example, generic view logic 930, specific point of view logic 940, and specific point of view data 950. In Figure 9, a user 910 interacts with business event handling logic and/or data 920 through a graphical user interface 900. The graphical user interface 900 may reside, for example, on a computer component having a display, a selection device, and a method of providing and selecting from a set of data entries on the display. While a display is discussed, it is to be appreciated that other communication techniques (e.g., audio, haptic) may be employed with interfaces associated with the systems and methods described herein.

To facilitate the user 910 interacting with the business event handling 920, the graphical user interface 900 acquires a set of data entries whose members represent one or more of a generic point of view logic operation, a specific point of view logic operation, and/or a specific point of view data operation. After retrieving the set of data entries, the graphical user interface 900 displays the set of entries on the display. Thus, one or more generic view logic entries 930, specific point of view logic entries 940, and/or specific point of view data entries 950 can be displayed on the graphical user interface 900. After display, the graphical user interface 900 receives a data entry selection signal that indicates that the user 910 has made a selection concerning one of the generic view logic entries 930, the

specific point of view logic entries 940, and the specific point of view data entries 950. In response to this data entry selection signal, the graphical user interface 900 initiates an operation associated with the selected data entry which facilitates the user 910 interacting with the business event handling 920.

Figure 10 illustrates an example API 1000 that facilitates accessing, for example, generic view logic 1040, specific point of view logic 1050, and specific point of view data 1060. The application programming interface (API) 1000 is illustrated providing access to business event handling 1030. The API 1000 can be employed, for example, by programmers 1010 and/or processes 1020 to gain access to processing performed in association with the business event handling 1030. For example, a programmer 1010 can write a program to perform selected business event handling 1030, including, but not limited to, generic view processing and specific point of view processing. Performing the generic processing can be facilitated by using the generic view logic 1040 component of the API 1000. Similarly, performing the specific point of view processing can be facilitated by using the specific point of view logic 1050 component of the API 1000 and the specific point of view data accessing component 1060 of the API 1000.

Processes 1020 may also interact with business event handling 1030 through the API 1000. Some processes 1020 may employ the generic view logic component 1040 to facilitate performing business event handling 1030 from the generic point of view. For example, regardless of the role, context, relationship, point in time (e.g., point of view), of an entity, there may be business event handling logic that will be performed. Thus, the process 1020 may employ the generic view logic component 1040 to facilitate this type of generic processing. However, depending on the point of view of an entity, certain logic may be performed in a context, role, time, relationship, dependent manner. Thus, a process 1020 may employ the specific point of view logic component 1050 and/or the specific point of view data component 1060 to facilitate this type of processing.

In one example API 1000, a first interface 1040 passes data and/or control associated with generic view logic. A second interface 1050 passes data and/or control associated with specific point of view logic. Similarly, a third interface 1060 passes data and/or control associated with specific point of view data.

What has been described above includes several examples. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the systems, methods, and computer readable media associated with

business process policy objects. However, one of ordinary skill in the art may recognize that further combinations and permutations are possible. Accordingly, this application is intended to embrace such alterations, modifications, and variations that fall within the scope of the appended claims. Furthermore, to the extent that the term "includes" is employed in the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as that term is interpreted when employed as a transitional word in a claim.

CLAIMS

What is claimed is:

1. A system for abstracting point of view specific logic out of a business process policy object, comprising:

a business process policy object that implements a first business intelligence for handling a business event from a generic point of view;

a policy definition object, related to the business process policy object, that implements a second business intelligence for handling the business event from a specific point of view; and

a user policy object, associated with the policy definition object, that stores a first data that facilitates executing the second business intelligence in the policy definition object.

2. The system of claim 1, where the business event is one of, a reference event, a change event, a threshold event, a task completion event, and a task failure event.

3. The system of claim 1, where the first business intelligence comprises at least one of, visualizing an event, reporting out an event, correlating one or more events, processing an event, and predicting one or more events.

4. The system of claim 1, where the second business intelligence comprises at least one of, visualizing an event, reporting out an event, correlating one or more events, processing an event, and predicting one or more events.

5. The system of claim 1, where the first data comprises one or more of, a threshold value, a key performance indicator identifier, a business process policy object identifier, a policy definition object identifier, a point of view specific policy value, and a context specific policy value.

6. The system of claim 5, where the first data is sharable on one or more of, a role based basis, and a user based basis.

7. The system of claim 5, where the first data is modifiable by one or more of, a graphical user interface, a command line interface, an application programming interface, and an object oriented message.

8. The system of claim 1, where the specific point of view is one or more of, a user specific point of view, a role specific point of view, a context specific point of view, a time related point of view, and a relationship based point of view.

9. The system of claim 1, comprising a policy manager for dynamically defining and instantiating a policy definition object.

10. The system of claim 9, where the policy manager manages a user context override for a threshold value to apply during execution of one or more of, the first business intelligence, and the second business intelligence.

11. The system of claim 1, comprising a business process policy manager for managing the life cycle of a policy definition object via a life cycle management request.

12. The system of claim 11, where the life cycle management request is one or more of, a policy definition object create request, a policy definition object delete request, a policy definition object update request, and a policy definition object query request.

13. A computer readable medium storing computer executable components of the system of claim 1.

14. A method for separating point of view specific logic from generic view logic, comprising:

creating a business process policy logic for handling one or more business events;

abstracting from the business process policy logic a point of view specific logic and a generic view logic;

implementing the generic view logic in a business process policy object;

implementing the point of view specific logic in a policy definition object; and

storing a first data in a user policy object to facilitate executing the point of view specific logic.

15. The method of claim 14, comprising registering the policy definition object with a policy dispatcher to expose at least one of, the generic view logic, and the point of view specific logic.

16. The method of claim 15, comprising:

receiving a business event;

mapping the business event to a policy definition object; and

executing a business process policy object in the context of one or more user policy objects related to the policy definition object to which the business event was mapped.

17. The method of claim 14, comprising managing the life cycle of a policy definition object by a life cycle management request.

18. The method of claim 17, where the life cycle management request is one or more of, a policy definition object create request, a policy definition object delete request, a policy definition object update request, and policy definition object query request.

19. A computer readable medium storing computer executable instructions operable to perform computer executable aspects of the method of claim 14.

20. A set of application programming interfaces embodied on a computer readable medium for execution by a computer component in conjunction with a business event handling logic, comprising:

a first interface for managing a generic view logic in a business process policy object;

a second interface for managing a specific point of view logic in a policy definition object; and

a third interface for manipulating a specific point of view data in a user policy object.

21. The application programming interfaces of claim 20, where managing a generic view logic comprises one or more of, establishing, invoking, and removing a logic.

22. The APIs of claim 20, where managing a specific point of view logic comprises one or more of, establishing, invoking, and removing a logic.

23. The APIs of claim 20, where manipulating a specific point of view data comprises one or more of, inserting a data value, deleting a data value, querying a data value, and updating a data value.

24. In a computer system having a graphical user interface comprising a display and a selection device, a method of providing and selecting from a set of data entries on the display, the method comprising:

retrieving a set of data entries, each of the data entries representing one of a generic point of view logic operation, a specific point of view logic operation, and a specific point of view data operation;

displaying the set of entries on the display;

receiving a data entry selection signal indicative of the selection device selecting a selected data entry; and

in response to the data entry selection signal, initiating an operation associated with the selected data entry.

25. A system for performing context specific business process policy processing for one or more different contexts for a single business event, comprising:

means for executing a generic business process policy intelligence;

means for executing a point of view specific business process policy intelligence; and

means for relating a context to the generic business process policy intelligence and the point of view specific business process policy intelligence to facilitate executing the generic business process policy intelligence in association with one or more point of view specific business process policy intelligences.