Business-driven application security: From modeling to managing secure applications

N. Nagaratnam
A. Nadalin
M. Hondo
M. McIntosh
P. Austel

Business-driven development and management of secure applications and solutions is emerging as a key requirement in the realization of an on demand enterprise. In a given enterprise, individuals acting in various roles contribute to the modeling, development, deployment, and management of the security aspects of a business application. We look at the business-application life cycle and propose a policy-driven approach overlaid on a model-driven paradigm for addressing security requirements. Our approach suggests that security policies are to be modeled using policies and rule templates associated with business processes and models, designed and implemented through infrastructure-managed or application-managed environments based on modeled artifacts, deployed into an infrastructure and potentially customized to meet the security requirements of the consumer, and monitored and managed to reflect a consistent set of policies across the enterprise and all layers of its application infrastructure. We use a pragmatic approach to identify intersection points between the platform-independent modeling of security policies and their concrete articulation and enforcement. This approach offers a way to manage and monitor systems behavior for adherence and compliance to policies. Monitoring may be enabled through both information technology (IT) and business dashboards. Systematic approaches to connect business artifacts to implementation artifacts help implement business policies in system implementations. Best practices and security usage patterns influence the design of reusable and customizable templates. Because interoperability and portability are important in service-oriented architecture (SOA) environments, we list enhancements to standards (e.g., Business Process Execution Language [BPEL], Unified Modeling Language™ [UML®]) that must be addressed to achieve an effective life cycle.

INTRODUCTION

Enterprises must continually adapt to changes that occur due to business, political, or technological challenges. These on demand businesses require

integration of people, information, and processes in order to conduct business in real time. Meeting the requirements of such a dynamic environment requires leveraging business-to-business (B2B) partnerships and outsourced services by enabling enhanced integration between business processes. For example, supply chain integration of manufacturers and distributors requires deeper examination of sales forecasts, production scheduling, product configuration, and inventory management.

Recently, government requirements for accountability of business practices and information management have transformed security concerns from an isolated piece of the information technology (IT) puzzle into an important and far-reaching business issue that must be addressed. It is no longer sufficient to delegate responsibility to the IT organization alone. Doing so may lead to fragmented business and IT plans along with misallocation and inefficient use of already scarce technology resources.

To satisfy the new demands of a changing marketplace, the industry must adopt a fundamental change in the way application and system integration is accomplished. This change requires an infrastructure that supports loose coupling of intra- and inter-enterprise information among widely disparate application designs, operating systems, databases, and application programming interfaces (APIs). In order to efficiently integrate the varied set of applications and platforms that make up the information technology (IT) infrastructure of these enterprises, the enterprises are beginning to realize the value of a service-oriented architecture (SOA) and to refactor their applications into loosely coupled services. For an enterprise to be a secure on demand business, the enterprise infrastructure must be flexible and customizable to reflect new requirements and regulations. To provide such flexibility, the enterprise should not hardwire (permanently fix) its policies into the infrastructure, but instead allow the security model of the enterprise to be implemented through a policy-driven infrastructure. This is no simple task.

A step-wise approach to model, design, implement, deploy, and manage secure applications by using policies to reflect the business goals and to abide by constraints imposed through regulations (industry, federal, etc.), corporate security policies, and business trust relationships allows organizations to unlock the true value of IT security. We outline the importance of using a business-driven development methodology.[1] This methodology takes advantage of a business-process-modeling and Model Driven Architecture** (MDA**)[2,3] approach to separate the platform-independent model of the application architecture from the underlying implementation technology and platform. The value proposition of MDA is about enabling "automation and abstraction using open standards."[4] Additionally, a policy-driven approach to MDA acts as a powerful mechanism for management of security policies throughout the application life cycle.

We propose an approach to efficiently model, build, and manage secure enterprise applications in a dynamic environment. The process starts with the modeling of businesses by collecting business drivers and business requirements. The business model helps build an understanding of the business implications of application design and deployment decisions. This process encourages business analysts as well as security architects to formally explore the security aspects throughout the application life cycle. Business process modeling may be used to capture the information flow and process elements required for new applications. The business process model helps build an understanding of any additional tooling and deployment support that may be required to handle application development and management. Each enterprise and each application requires different amounts of involvement by analysts pertaining to its line of business and by architects and developers with respect to where security requirements enter the application life cycle.

Managing a secure on demand business is an ongoing learning experience. We start with an assumption that incorporating security planning into a company's overall corporate strategy and business process not only helps mitigate risks but also helps position an organization for long-term growth. Using a business-driven security-policy-management framework that starts with core business objectives allows businesses to identify suitable security mechanisms.

We begin with an overall discussion of the application life-cycle phases and a set of business roles. Individuals performing these roles perform

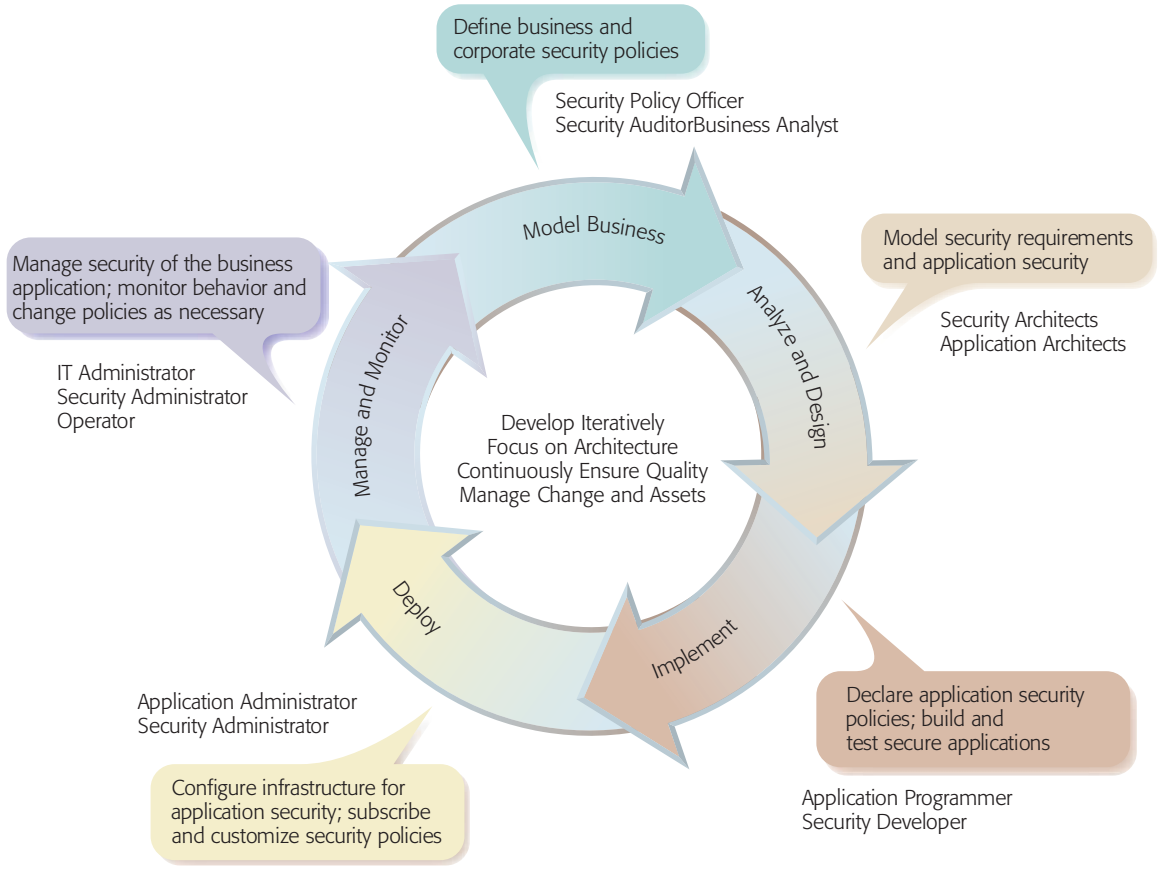

Figure 1
Refining and defining security policies in a business-driven development process

the tasks within the life-cycle phases in order to accomplish the business goals. Then, each of the phases in the application life cycle are discussed in detail. The details for each phase include the positioning of the phase in the overall life cycle, the kind of inputs and outputs that are relevant to a tooling application in a given phase, the tools and technologies that are required to accomplish the approach, and any standardization that is necessary in relevant technical approaches. We use an example throughout that illustrates how a higher level business policy is transformed, implemented, enforced, managed, and monitored in the process.

APPLICATION LIFE CYCLE AND ROLES

To enable a business so that its processes and applications are flexible, one must start by expecting changes—both to process and application logic and to the policies associated with them. The concept of change must be part of the conceptualization of the business idea. One may start by modeling the business, including business processes, organizations, system assets, and topology. A second pass should be made to identify areas in which growth or change is anticipated.

Software applications are designed and built in new ways to enable and automate business processes. As depicted in ***Figure 1***, the life cycle of an application built around a business-driven development methodology includes the following phases:

- *Model business*—Modeling the business process independent of whether the activities of which it is comprised are based on software,
- *Analyze and design*—Application modeling in a platform-independent manner,
- *Implement*—Implementing and testing applications on a chosen platform,
- *Deploy*—Installing an application within an infrastructure and subscribing for usage by service consumers,

Table 1 Security-defining roles in an organization

Organization	Roles
Business Strategy and Decision Making	Chief security officer, security policy officer, security architect, security auditor, business analyst
Development	Application programmer, identity/security developer
Operations and Administration	Security administrator, system/application administrator, operator

- *Manage and monitor*—Managing application configurations and monitoring application behavior.

Such an approach involves iterative development while focusing on consistent architecture. The need to continuously ensure quality of development software and ability to manage changes and assets is to be taken into account. Applications with these qualities help build business solutions consisting of new applications as well as existing application assets. When working with existing assets, which may be deployed on different platforms and environments, an SOA architectural pattern helps to bridge platform-specific nuances and abstract out service functionality. The modeling phase identifies services that are independent of the implementation phase. A service veneer may be developed to connect to an existing implementation, or an entirely new application may be developed. The primary benefit of this approach is the agility to respond to changing business requirements while the underlying technology infrastructure evolves at its own pace.

Understanding enterprise roles and responsibilities

Individuals acting in roles within an organization take on responsibilities within that organization. They make decisions to ensure that the technology and implementation meet the business requirements, and they increasingly use tools to efficiently execute the security plan. Thus, tool support is very important to help individuals acting in various roles to efficiently fulfill their assigned responsibilities. These roles also typically represent the organizational structure of the business. A sample list of these roles is depicted in *Table 1*.

If the life-cycle model is to be successful, it is important to understand the roles that individuals will perform during the application life cycle and the tasks they must perform. Depending on the responsibilities assigned to each role and which part of the business they represent, the associated tasks may vary. A set of roles is defined to manage security and business policies.

As shown in Figure 1, certain roles in an organization contribute toward creating, defining, refining, or managing security policies throughout the life cycle. They include the following:

- Corporate security officers and equivalent executives defining corporate security policies and outlining regulations with which the business must comply,
- Business analysts working with security policy officers, translating corporate policies in terms of a business vocabulary and a business process during the business-process-modeling phase and providing a set of choices to be customized,
- Application architects and security architects modeling the security and access policies in the model (based on the choices provided by a business analyst) during the application-modeling phase,
- Application developers factoring in these security policies by declaring these requirements for the infrastructure to enforce, or when infrastructure support is not sufficient, implementing them in their applications; or application deployers installing the applications and working with security administrators to configure these applications and the security policies as relevant to the deployed environment,
- IT and security administrators managing the security policies throughout a set of applications and an infrastructure to meet the requirements that may continue to change over time,
- Operators monitoring the system behavior and detecting situations that are potential security threats and feeding that back to administrators to make any changes necessary for the application infrastructure to adhere to the goals; similarly, a

business analyst viewing business dashboards to observe the impact to the business due to certain system security events.

It is significant to observe that security policies are specified and refined throughout the life cycle, undergoing transformations from one phase to the next. In the current state of the art, it may be realistic to expect this to happen in a unidirectional manner—from modeling to monitoring and management. In order to accommodate bidirectional flow within the life cycle, traceability support and sophisticated transformation support between artifacts from one phase to the next phase is required. For example, for Web service development based on the Java** platform, conversion from the Web Services Definition Language (WSDL) to Java as well as from Java to WSDL must be possible. As long as the transformations are symmetric and consistent, the potential exists for iteration in both directions; therefore, any possible iteration required from one phase to a previous phase (e.g., from implementation back to modeling) should be part of the evolution of tooling.

Authoring corporate security policies

As part of formulating a security strategy and authoring corporate security policies, the chief security officer is responsible for knowing the set of legal, business, and financial policies to which the organization must comply. It is part of the responsibility of the individual playing that role to articulate these requirements to the organization. Often this is done through the authoring of documents that contain some level of detail or directives about the requirements in natural language.

At this level, the business security policies are usually goals and guidelines and are often expressed in corporate documents, not development artifacts. The process of identification is part of the security risk assessment and uses a methodology such as OCTAVE** (Operationally Critical Threat, Asset, and Vulnerability Evaluation)[5] or the FISMA (Federal Information Security Management Act)[6] guidance from NIST (National Institute for Standards and Technology).

Within the staff of the security office, there are security policy officers (SPOs) and security architects. SPOs and security architects usually take on the responsibility of translating the corporate policies and guidelines and defining the security policies in a form that may be used by the rest of the organization. The task of translation may involve translating policies written in natural language expressions into forms that are usable by tools. Such corporate policies could be creating or annotating a business requirement by using a security vocabulary, as appropriate. The SPO may use a tool that transforms the natural language expressions into machine- or tool-usable form for an efficient development life cycle. For example, a policy stating that "travel agents may view and change the traveler's itinerary" may be transformed into XML (Extensible Markup Language) fragments as shown below. Such XML fragments may then be attached to modeling artifacts later in the life cycle of application development.

```
<role> Travel agent </role>
<actions>
  <action> view </action>
  <action> change </action>
</actions>
<resource> traveler itinerary </resource>
```

MODELING SECURE BUSINESS PROCESSES

SOA design patterns help an enterprise to identify and eliminate redundant implementations of common business processes by facilitating coordinated, consistent, and efficient implementation and management of enterprise-wide policies. Business process modeling provides a means for business analysts to formally define a process to reflect the inner workings of a business. Formalizing this process with a standard methodology by using business process modeling helps to efficiently implement the idea. For example, the IBM WebSphere* Business Integrator (WBI) Modeler is a tool that allows a visual model of a business process to be built, resulting in both a visual representation and potentially some set of artifacts representing the business process (i.e., WSDL, policy templates).

In the process of modeling business processes, some activities may be automated through the implementation of software components. This automation may require a transformation to map a business function into a new application component, into a new programming element, or into a system element that previously existed in the deployment environment but which is being retasked. For example, a transformation from WBI Modeler to

Business Process Execution Language (BPEL)[7] may be further transformed into application artifacts represented by UML** (Unified Modeling Language**), which are then manipulated using tools such as the Rational* Solution Modeler or Rational Solution Architect.

Depending on tool sophistication, traceability, and point of entry into the life cycle for a particular business, the transformation may be one way, from the business process visualization to business service or activity design, or bidirectional, whereby changes in application modeling reflect back in the business process models. Nonetheless, experience indicates that business analysts do not want random changes to the business process to occur based on some change to implementation artifacts.

Business process modeling is an ideal time in the life cycle of a business solution and a repeatable process mechanism to begin to capture the business security requirements that address any security concerns that relate to the business, thus performing security risk assessment. A business analyst working with an SPO is typically involved at this point. The business analyst (e.g., with help from the SPO) determines which policies apply to a given business in the context of the business process. For example, in a travel business process, the business analyst might model the requirement to control authorized access to a business activity (such as a travel reservation) and to ensure that the business information flow is protected for confidentiality. The business security requirements can be defined within the business process models. These models provide a reference that may be used by enterprise compliance officers, such as security auditors, to verify and monitor adherence to enterprise security policies.

Among the tasks performed during this phase, a business analyst may specify the requirements for the artifacts of a business process that are generated during a particular phase of the life cycle. Note that business analysts will not likely define these requirements in terms of security technology like secure sockets layer (SSL), message security, or encryption, but they will typically define the broad goals and ranges of offerings and thus, allow for flexible implementations that meet these objectives. As shown in *Figure 2*, a business analyst may specify the requirement to allow authorized access for travel agents to view and change itinerary. Also, the analyst can describe these requirements using security categories. For example, a travel service consumer may be able to select an appropriate service level, which includes High or Medium security as part of the Platinum or Gold offerings, respectively.

Individual policies pertaining to specific business process elements may be defined at this stage. These policies would reflect objectives from various sources, including regulations, compliance, industry, competition, or business-specific objectives. Mechanisms should be available to enable these policies to be defined and associated with appropriate points in the business process that are constrained by these policies. Such policies will normally be defined in terms of business domain vocabulary and may be derived from sources established by other agents in the business (e.g., Chief Privacy Officer), as appropriate.

To facilitate reuse from existing, well-known objectives that must be met, policy authoring should leverage policy templates that reflect well-known policies and rules. For example, a travel industry regulation states that a travel agent may only look at customer information for the purposes of ticketing. During business process modeling, one should be able to review these policies and contextually apply them to the appropriate business objects being modeled. For example, what constitutes a ticketing activity is specific to a given business process. Similarly, sensitive business objects may be tagged to be protected.

A business analyst may be cross-trained and also be responsible for interpreting security requirements (e.g., an audit of Sarbanes-Oxley[8] compliance) and modeling the point of enforcement within the process itself. For example, if the requirement articulated by an SPO is to audit all travel approvals, the business analyst may create an activity to perform an audit (after travel approval activity), and this may be modeled within several business processes.

In addition to modeling requirements and policies, one should be able to specify any key performance indicators (KPIs) (if they are known) that are to be met—which, in turn, must be monitored and managed by some other element in the workflow. From a security perspective, such monitorable aspects may be defined in terms of security

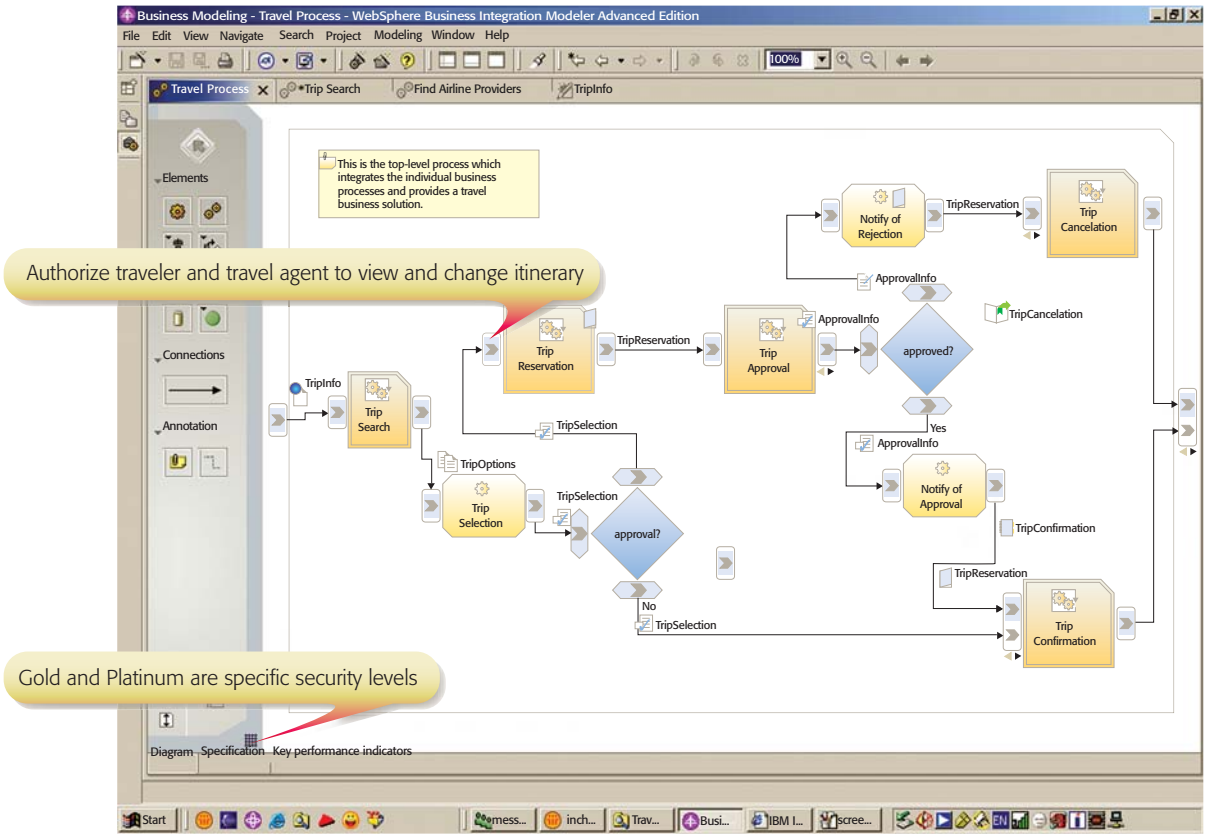

Figure 2
Business-level security policies in business process modeling

situations, intrusions, security breaches, or policies pertaining to accessing sensitive information. Information categorization (e.g., highly confidential, informational) during modeling may help clarify what level of security enforcement is required.

In most currently deployed applications these types of enterprise-level decisions and policies have been codified and enforced redundantly and inconsistently in individual applications under the auspices of application programmers. When a business-driven approach is used, business process models that capture the intent of the business have policies associated with them to indicate these types of requirements and decisions. These policies may then be used to drive the behavior of common service components that implement important business logic. The business may then build a coordinated administration and configuration design. This helps ensure that enterprise-level policies reflecting the intent of the business stakeholders are accurately and consistently implemented across all applications in the enterprise. The enterprise may track any changes to policies and ensure that they are efficiently propagated.

Input

Typical security concerns that would be considered as part of the definition of a business vocabulary and reflected in related business process models would include the following:

- Controlled access to customer and employee business information,
- Monitoring and audit of security situations and events, including those related to storage, access, and retention,
- Confidentiality protection from inadvertent disclosure of information contained in or derived from data flows,
- Integrity of data in data flows or storage, protecting it from accidental or purposeful modification,
- Nonrepudiation of data origin and data receipt, including concerns related to authentication and audit controls,

- Trust of entities operating within business processes, including business partners, service providers, and users and system components, both internal and external (e.g. partners).

Output

The outputs from this phase typically include business process definitions reflecting high level business objectives, organization, roles, partner relationships, and process components. Included in this model would be elements denoting any business level security concerns. From these models, security policies may be generated to drive conformance by concrete instantiations of these model elements.

Standards and technologies

From an SOA perspective, the primary standard related to this phase is BPEL. At the time of this writing, BPEL extensions may be used to accomplish articulating policies; formalization of these extensions to profile policy definitions within BPEL is envisioned. There may be other mechanisms some tools may use; for example, one may model using tools such as IBM WBI Modeler, using standardized notations such as Business Process Model Notation** (BPMN**),[9] to capture the business process which ultimately may be exported to a BPEL workflow. Similar to BPEL requirements, the ability to attach policies and rules to BPMN is useful. One may use the annotations capability in BPMN as well as BPEL extensibility mechanisms.[7] If BPEL is used to represent the enterprise business processes, it is important to represent security policies and constraints by using first-class notations and artifacts. In addition to corporate policies, industry- and locality-specific regulations will apply to a business process, and therefore, tools should allow for working with profiles of these policies targeted to a specific industry. Part of any industry-specific standardization of a BPEL pattern should be a recommendation to profile the policies of the specific industry as an input pattern for individual businesses to exploit.

An emerging standard business process definition metamodel (BPDM)[10] being developed at Object Management Group (OMG) and led by IBM and other BPM vendors defines how MDA concepts and standards may be used to support the definition of

- the business process model (with appropriate links to business rules as well as organizational models),
- support for multiple visual notations mapped to this model (i.e., BPMN for a business analyst, UML for a IT architect and security architect), and
- transformations (mappings) from the metamodel to Web Services artifacts, such as BPEL, WSDL, and XML schema.

This initiative exemplifies how elements of model-driven security architecture may be integrated with business process modeling and business performance management.

The requirements may go beyond simple security attribute notations to include cross-level (business modeling to application modeling) modeling, policy generation and validation, traceability, discovery and validation of business processes through analysis of deployed implementation artifacts, and a useful visual method for depicting a model of security patterns and practices.

DESIGNING SECURE BUSINESS APPLICATIONS

For application modeling and design, system architects typically employ UML[11] and its profiles to model the concerns of the service domain. Toward this, the Meta Object Facility (MOF**) and its implementation are an enabling technology foundation to achieve these goals.[12]

The security architects should keep implementation-specific security concerns out of the application model. At the same time, security should not be an afterthought. One should be able to capture generic security requirements and constraints in a model in a consistent fashion. Given the complexity of security implementations and its impact on system design and deployment, security requirements and policies place additional constraints and requirements on the IT infrastructure supporting the services. Security design patterns are becoming increasingly popular and useful at this stage. If weaknesses are discovered, they should be given to the system architect for alternative approaches.

Applying the modeling concepts to security, we assume that users define abstract security intentions at a higher level, and then they are refined at a lower concrete level. UML may be used at the lower three levels. It is not clear how to model with UML at the strategy level; therefore, we begin by addressing security at the operation level. We summarize what is described for security at each level as follows:

Table 2 Primitives to capture security constraints

Primitive	Description
audit	Denotes that the specified communication is to be audited. One assumes that the audit will contain both the identity of all parties and any data communicated between the parties
authenticate	Denotes that a party that has to be authenticated in the scope of a given context
authorize	Denotes that a communication between two parties must ensure that the requestor is authorized to perform the request
confidentiality	Denotes that the marked information should be treated as confidential and that all reasonable effort (considering technical implications) should be made to ensure the data remains protected against unauthorized viewing
integrity	Denotes that the marked information is guaranteed to reach the recipient unaltered during transit (e.g. includes the digital signatures of parties related to the document)

- *Platform-independent model*—Only security intentions are added to the business process models. Primitives for security policies should be abstract enough so that business users understand them. Policies may be implemented on any software platform.
- *Platform-specific profiles*—Security requirements derived from the intentions are added to the executable artifacts such as the J2EE** (Java 2 Enterprise Edition) application packages. While security may be included in application code, we assume that security requirements are described with policy in the configuration of application packages; for example, the deployment descriptor (DD) for J2EE.[13,14]
- *Protocol- and technology-specific profiles*—The security requirements at the execution level are bound to specific message protocols, security infrastructures, and technologies available to the application hosting environment. For example, we may consider security mechanisms, such as PKI and Kerberos, and cryptographic algorithms for signatures and encryptions.

In order to address business-level-security intent in a manner that is easy to understand, even for business users, we adapt and modify the security primitives proposed in Reference 15. These primitives are abstract enough for our purposes and may be represented in UML. Examples of the security policy elements or primitives can be found in *Table 2*.

The primitives in the table may be represented in UML by using model artifacts like UML stereotypes, constraints, and so forth. It is useful to have a standardized security profile in UML that defines some of the security primitives in terms of UML stereotypes. As shown in ***Figure 3***, one can model authorization policy as a UML stereotype with a set of roles about who can access a Reservation object. Additional conditions about time of access, type of access, and purpose can be encapsulated. Policies are also applicable to invocations as shown—to ensure a `makeReservation()` method invocation for confidentiality. Such an approach helps an application designer capture the security policies at the application-modeling phase.

Input

The inputs to the design phase are policies and rules which were transformed from business process models into UML artifacts and security requirements applicable to the application design. Depending on the design details, these requirements may include platform-dependent requirements (e.g., based on the J2EE security model).

Output

The outputs from the application modeling and design phase typically include model specifications (UML) and code-generated skeletons of implementations (e.g., J2EE application components). Included in the model and implementation artifacts would be declarative descriptors denoting design-level security requirements and possibly codified policies.

Standards and technologies

As discussed earlier, there are several possible approaches to model security policies in UML using

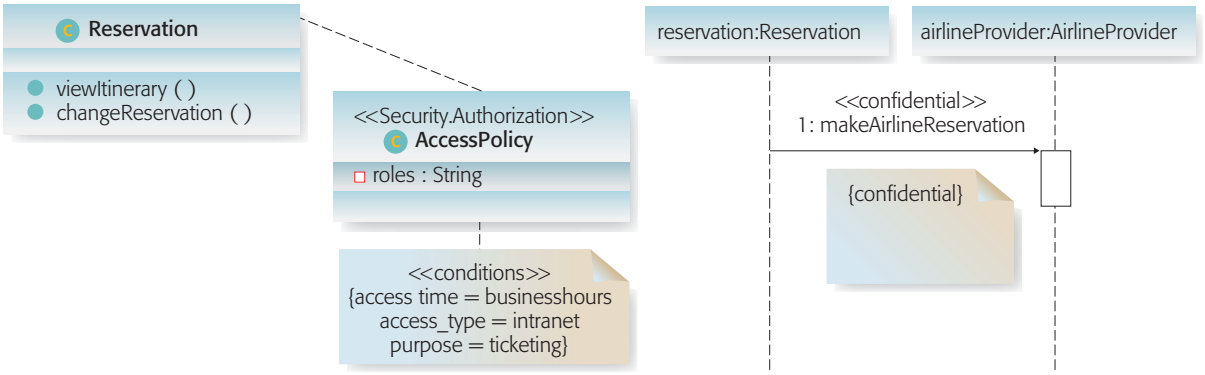

Figure 3
Applying security policies to application models

profiles, metamodels, constraints, or other mechanisms. Previous publications describe some of these approaches.[15,16,17] When a stereotype-based approach is employed, these security stereotypes may be transformed into appropriate deployment artifacts (e.g., Enterprise JavaBeans** [EJB**] deployment descriptors). Specific profiles may be applied and artifacts generated through tooling. In other cases, when an application manages its own authorization decisions, standardized profiles may help identify callouts to security decision points to evaluate and make decisions, which the applications then enforce.

IMPLEMENTING SECURE SOLUTIONS

Depending on the environment and existing functionality, implementation may entail building new application components or reusing existing components and applications with appropriate wrappers and configuration supporting the new functionality. In order to accommodate the changing environment and SOA, these components may be implemented as Web services. This implies an interface defined in Web Services Description Language (WSDL) that enables access from other services. The service implementation depends on the underlying platform. For instance, when hosted on a J2EE[18] platform, the Web service implementation may be a J2EE application comprised of EJBs.

If the implementation phase follows the design phase, the UML model artifacts may be used to generate skeleton code (e.g., service components, EJBs) along with security policies that must be transformed from the earlier phase so that the Web services are secured.[19] A profile may be used to automate XSD (W3C** XML Schema definition language) and WSDL generation.[20]

If the implementation was targeted without any formal design phase with UML, then the application implementation must factor in various business requirements, corporate policies, and other considerations in the application code. Beginning to implement all the details in this phase is a significant burden on the developer; whereas, starting from an earlier phase eases the task of the developer in dealing with all of these security policies.

When designing a solution, a developer should identify the components to be integrated to create the solution. Because our goal is to address an environment based on SOA, we assume that the application is exposed by using a service description that consumers access.

Implementation considerations

From a security perspective, the implementation phase requires decisions to be made about which components are responsible for enforcing security policies (e.g., infrastructure, application) and which information must be made available to requestors. In addition to the operational aspects, some of the design-time policy information (e.g., J2EE deployment descriptors) may be used to help manage the application. One of the key implementation decisions is whether business requirements are best met by implementation of the security model in the infrastructure or by codifying security enforcement into each application. Another dimension to implementation that should be considered is the degree of variability of the invocation of the service. Is

flexibility given to consumers (i.e., are there choices that are made available for consumers to customize during subscription)? Lastly, when implementing secure solutions, the concept of security engineering, an engineering methodology to build secure applications, must be considered. An extension to Rational Unified Process* (RUP*)[21] may help enable this approach in a systematic way.

Security engineering

Security engineering in development of secure solutions involves following well-defined patterns and best practices in implementing application components so that the application does what the designers and users expect it to. A part of the task is assessing the risk inherent in each of these components and designing and implementing in such a way that we avoid opening them up to known vulnerabilities (e.g., efficient memory management, avoiding covert channels). Tooling support and code reviews are important to improve good practice in security engineering and to do no harm or help minimize harm to the environment in which these solutions may be deployed.

From the viewpoint of security engineering, Java is well-positioned as a development language due to its built-in memory management capabilities, platform independence and security sandbox model that encourages the practice of building secure solutions. The Java 2 Security model helps define security policies pertaining to the access of applications to resources external to the application (e.g., permissions to access files, sockets, threads, etc.).

Infrastructure-managed vs application-managed security

The application-modeling phase may have introduced decisions about infrastructure-managed vs application-managed security. For example, one may choose to attach security policies and requirements to UML artifacts (classes, objects, activity diagrams, etc.). In this case, these requirements are meta-data attached to these artifacts. During the implementation phase, more information about the application platform (e.g., J2EE, Microsoft .NET**) is likely to be available, and decisions either to let the infrastructure handle security or to codify security in the application may be made at this point. Thus, a security architect should design some standardized patterns so that these decisions are made in a consistent manner.

In cases where the infrastructure-managed approach is taken, policies attached to modeled artifacts are transformed into platform-specific policies (e.g., UML requirements to J2EE deployment descriptors). In cases where the application-managed approach is taken, security enforcement is performed by the application and the appropriate security callouts must be implemented. Tools can help translate any callouts in application models into platform-specific implementations. For example, the callout `authenticate()` can be transformed to `loginContext.login()` by using Java Authentication and Authorization Service (JAAS),[22] and `authorize()` can be transformed to `policy.implies()` by using authorization checks based on Java Authorization Contract for Containers (JACC).[23] In either of these cases, platform-specific transforms and profiles are necessary to help implement the intended approach and thus enforce the security policies at desired enforcement points. Such profiles help improve the security of the application by reducing the number of security-specific application programming interfaces (APIs) and technologies that the developer must use (thus reducing coding and improving the quality of the application software). Based on various usage patterns, the tools may provide certain well-known implementations based on guidelines, best practices, and so forth, in which case, a developer works with patterns instead of making decisions about technology. Such patterns may help transform the intent of application modeling where the platform (e.g., J2EE) may be known, but a finer selection of technologies (as relevant to the application) may be done only during the implementation phase (e.g., using JAAS callout for authentication or calling out to a security framework managed by the IT department of the enterprise).

Deciding the flow of authentication and authorization becomes a key part of the implementation phase because the new solution must be implemented within the context of the existing IT organization. All of these initial security policies for the applications may be defined by using deployment artifacts (e.g., deployment descriptors for J2EE applications). Declarations of policies should be abstracted and used to provide high-level policy requirements that may be refined later in the deployment phase, taking into account any implementation constraints.

In the case of authorization, policy abstractions may be achieved by defining application roles as a collection of permissions which allow actions on resources. For example, a travel application may declare that the `view()` or `change()` methods on ReservationBean may be accessed by a user acting in the TravelAgent role. (i.e., TravelAgent is a name of a role that may be used at implementation time to identify what may be done by a travel agent in terms of a set of permissions to invoke those methods on the respective EJBs) as shown in the code below:

```
<method-permission>
<role-name>TravelAgent</role-name>
<method>
  <ejb-name>ReservationBean</ejb-name>
<method-permission>
<role-name>TravelAgent</role-name>
<method>
  <ejb-name>ReservationBean</ejb-name>
  <method-name> view</method-name>
  <method-name> change</method-name>
</method>
</method-permission>
```

What is not likely associated during the implementation phase is the specific decision of which users are able to act as a TravelAgent. Assignment of users to roles is typically performed during the deployment phase and managed during the lifetime of the application.

These authorization policies may not be known to the individual requestor, and the business policies should articulate when and to whom these internal policies are revealed.

Designing authorization enforcement points is another important task. In some organizations, it is left to the infrastructure to enforce these authorization policies as message requests are processed by the infrastructure. In other cases, when applications require more control with security, applications have to explicitly perform the authentication, establishing and maintaining context and enforcing authorization. There are hybrid cases when the deployment environment provides support, where an application may use the infrastructure to perform authentication and security-context establishment, and may use the authenticated user identity to perform its authorization decisions. For instance, user identity is established by a J2EE runtime after authentication. An application may use standard APIs like `isCallerInRole()` to verify that an invoking user identity has been granted the ability to act in a specific role. In cases where identity is propagated downstream as part of invoking another service, the calling application component may use APIs like `getCallerPrincipal()` to retrieve information about the calling principal identity.

Flexibility of choice

In cases where certain requirements or constraints (on the access to the service itself—including authentication, integrity, and confidentiality requirements) should be made known to a requesting client runtime, an organization may be capable of providing a range of options to serve a wide variety of client runtimes (e.g., browser clients, non-browser clients, PDA (Personal Digital Assistant) thin clients, etc.). In this case, policies may be published that declare the requirement for a requestor runtime to ensure message confidentiality and provide some evidence of the identity of the requesting user (i.e., userid/password or a certificate). In the case of authentication, policy abstractions communicate a range of alternatives that include the types of credentials that must be presented or limits on the list of trusted authorities for issuing credentials.

For instance, a TravelService Web service may declare its intent to require certain token types and confidentiality requirements. Depending on the implementation, it may declare its intent in terms of appropriate descriptors. Tools may in turn generate necessary machine-level details (e.g., a WS-Policy expression) as shown in this code:

```
<wsp:Policy>
<wsp:ExactlyOne>
<wsse:SecurityToken> username </>
<wsse:SecurityToken> x509 </>
</ExactlyOne>
<wsp:AttributeCert Issuer=TravelOrg >
   wsp:optional=true
<wsse:Confidentiality>
  <wsse:Issuer=Verisign>
  :
</wsse:Confidentiality>
</wsp:Policy>
```

Input

Input to the implementation phase includes information from the application-modeling phase (e.g.,

UML class diagrams, activity diagrams). In the case of UML, the input may include application classes that represent interface abstraction for access, business object information representing data objects passed between method invocations, and activity diagrams. As identified in the application-modeling phase, any security policies associated with model artifacts in terms of access policies (constraints, including authorization that may be attached to classes, delegation options marked up in activity diagrams, etc.) may be done through UML (with necessary enhancements and extensions, as appropriate). The tools used during the implementation phase must be able to transform model artifacts into implementation entities (e.g., EJBs, service components). Relevant policy information must be transformed (e.g., J2EE deployment descriptors) and attached to these new implementation artifacts. This takes advantage of the fact that in J2EE there is a clear separation between the application code and associated roles and permissions in the deployment descriptors. The J2EE container enforces controlled access to the components, hiding the details from an application developer.

Output

Upon implementation and packaging, a solution includes declared policies that are associated with application components (e.g., in the case of J2EE components, in the deployment descriptors). Note that in addition to deployment descriptors for individual components, an aggregated deployment artifact is required to represent the collective policies of the integrated solution components. These declarative policies may be either in standard representations handled by the deployment infrastructure in the case of infrastructure-managed security or in some other format or rule interpreted by the application.

Policy technologies and standards

To support the infrastructure-managed security pattern, it is necessary to have technology support for declaring security policies to be enforced by the infrastructure. In addition to declaring requirements and intent in deployment descriptors, support for security is necessary in the infrastructure (e.g., fine-grained authorization support if fine-grained authorizations are articulated by the developer). Based on the information and decisions made in previous phases, verification of support from the infrastructure is either factored in through patterns in the earlier phases, or it now must be taken into account when implementing the application.

In order for applications to be designed to support flexibility for the consumers' choice, security policies should be defined in terms of business value and security levels. In addition to these categorizations, tools to help provide techniques, such as templates to define policies throughout the life cycle of a solution, are required so that abstract policies may be further refined through the life cycle. Tools that help transform these goals and requirements into technology-dependent artifacts (e.g., J2EE deployment descriptors) are required.

As noted earlier, one of the features required throughout the solution life cycle is tooling support for traceability. Even though traceability throughout the life cycle is possible, bidirectional traceability (e.g., changes reflected in implementation or during deployment in the application model) represents a potentially impractical challenge. When input policies from an application-modeling phase (e.g., UML) are transformed into implementation-specific policy descriptors (e.g., J2EE deployment descriptors), technology support is necessary to help correlate these policies as they have been transformed into new artifacts. To encourage traceability, an attempt should be made to reflect any changes to implementation policies back into application-model policies. In cases where application-model policies are implemented as application logic (or for a variety of other reasons including implementation platform characteristics), having such information captured in text as an annotation to the model, even through non-normative means, would be a useful addition.

When specifying policies, any user-facing tools should appropriately provide facilities to allow for policy configuration or updates from the user's perspective, hiding the details of the underlying runtime policy language (e.g., J2EE deployment descriptor file format, XACML (Extensible Access Control Markup Language) files). In a perfect world, tools should allow the validation of role assignments across intercomponent invocations within a given solution package so that inconsistencies in policy declarations could be identified early (e.g., through static analysis of solution code). Similar helper utilities and verification utilities are required, and

the security engineering team of an organization should help automate security best practices to help avoid security vulnerabilities. Developers should be provided the necessary tools to build user interface components for consumer-facing tasks, including subscription, security-level selection, security requirement specification, and the user login page.

DEPLOYING AND MANAGING SECURE SOLUTIONS

Modeled and implemented applications become "real" when they are deployed and become accessible to consumers. This may be viewed as the critical phase in every solution life cycle. Some solutions may be modeled, implemented, and deployed; some may be implemented directly and deployed; and some may be purchased, deployed, and managed. Regardless of various possible choices of arrival to this point, it is invariable that an enterprise will go through the phase that includes installing a solution, allowing consumers to subscribe to the solution, and managing the solution throughout its lifetime of service.

Solution deployers, IT administrators, and security administrators factor in the security-infrastructure and the enterprise-security requirements to ensure that the applications—when installed and deployed for access—are set to meet the requirements. At this point issues like heterogeneity and cross-platform support become concerns. Some of these policies may still be customizable if the services are offered for subscription by consumers. In that case, further refinement of applicable policies is performed by consumers upon subscription. Security administrators are responsible for configuring the systems and business applications for the security infrastructure they have deployed (e.g., corporate Lightweight Direct Access Protocol [LDAP] directory for user information and authentication or authorization provider such as IBM Tivoli AccessManager to help enforce access control both at entry into the intranet and at application and data access tiers). The security administrator continues to factor in changing requirements, threats, technology changes, and application behavior and manage policies to keep in synchronization with those changes. Policies of the applications and the infrastructure are updated and managed based on changing business and IT requirements and situations.

As shown in ***Figure 4***, deploying and managing solutions consists of a few subtasks: solution installation, subscription, and administration and management. Needless to say, policies are part of solution deployment and management. This section discusses how policies affect this phase of the life cycle.

Solution installation task

The task of installing includes binding the application to the enterprise environment in which it is deployed, making the solution accessible to consumers, and associating policies with the solution. This task may be viewed as a *solution installation* task. During this task, the life cycle roles that play a part include Solution Deployer, Security Administrator, IT Administrator, and Solution Administrator. They take into account the input available in this phase, which includes application-specific deployment information (e.g., deployment descriptors) and initial configuration information deduced through other policies made available to them. The roles use that information to bind the security policies to that solution. Depending on the configuration options that a solution may make available, relevant configuration is performed by these personnel. This is illustrated by the activities in the top right part of Figure 4. They use the solution-installation tools available to them to perform this task. For example, they may deploy a solution using WebSphere Business Integrator where the security options are customized using installation tools.

Additionally, in some distributed environments there may be an additional step. If multiple providers are possible, after the selection of appropriate providers (e.g., Tivoli AccessManager) is made to manage security decisions, the resource manager (e.g., application server runtime) distributes the relevant policies to these policy managers. The resource managers have the task of distributing those initial security policies to the policy managers in order to help improve integration and interoperability between different resource managers and policy managers. In addition to a standardized distribution protocol, there may be a requirement to make the policy information canonical. One may use standards like XACML[24] to express the policies, which may require transforming platform-specific policy descriptions (e.g., J2EE deployment descriptors) into XACML policies. For example, if the policy is further refined to elements of the application like portlets, such that a traveler can view itinerary and descendant pages: (Traveler, (View/Itinerary)), then

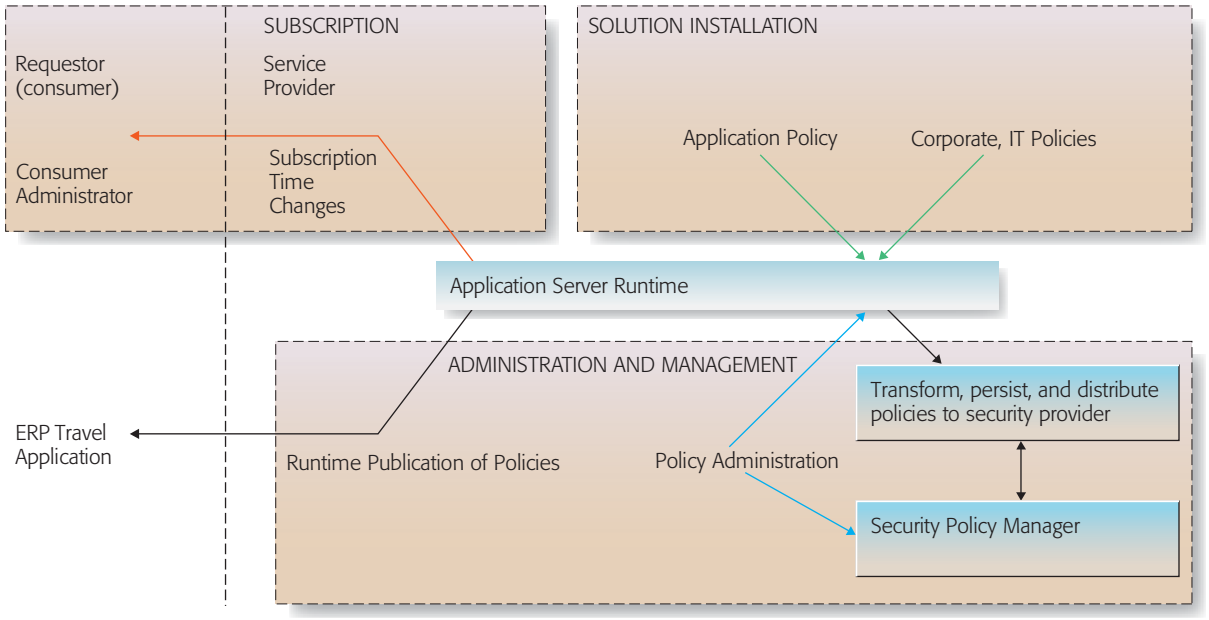

Figure 4
Deploying and managing security policies of an application.
The vertical dashed line separates external users and applications from the secure business application.

it can be transformed into XACML as shown in *Figure 5*.

Transformation and distribution activities are also illustrated in the bottom half of Figure 4 which shows policy distribution to policy orchestrators, policy managers, and so forth. There are cases where configuration information may be configured directly into the security policy managers, depending on the integration of management tasks between the hosting environment and security providers.

Subscription task

When a solution must be tailored to a given consumer, a *subscription* task is part of the process. During the subscription stage, consumers can customize the solution to their requirements and agreements (e.g., selecting a quality of service that is desired, selecting a service level, and so forth). When a solution may be customized during the subscription process, certain security variables may be allowed to be customized. Such information may be part of service-level offerings that are provided to a consumer. In the case of service-level offerings, the options may have been abstracted in terms of high-level constructs (e.g., High-Security part of Platinum service, Medium-Security part of Gold). Customizing what makes up High Security may be allowed (e.g., 128-bit SSL using Fabricam or Verisign certificates, using message-level security, using WS-Security, using tamper-proof audit) during the subscription phase. Customization is performed by individuals acting in appropriate roles: service-consumer business analyst, service-consumer security administrator and service-provider security administrator. These are illustrated by the arrow indicating subscription time changes in Figure 4.

Administration and management task

After a solution has been made available to a consumer, the solution must be managed and policies administered to reflect any changes that may happen during the lifetime of the solution. Changes to security policies include authorization policy changes (e.g., adding new roles that may access the resources or assigning roles to new user groups or users), user management changes (e.g., users assigned to additional user groups), or other changes including audit requirements and constraints like integrity or confidentiality. When administering security policies, it is necessary to adhere to changing corporate business-security policies and industry and government regulations and compliance requirements. In addition to these sources of change, another key input factor for change in policies is the discovery of vulnerabilities and new

```
<Policy PolicyId="P1"
        PolicyCombiningAlgoId=
   "path-more-specific-deny-overrides-with-propagation">
 <Target>
  <Subjects><Subject>
   <SubjectMatchMatchId="user-role-match">
    <SubjectAttributeDesignator AttributeId="subject-id"
                                DataType="string"/>
      <AttributeValue DataType="string">traveller
      </AttributeValue>
      <AttributeValue DataType="string">
          http://myUserRoleMapping</AttributeValue>
    </SubjectMatch>
  </Subject></Subjects>
  <Resources><AnyResource/></Resources>
  <Actions><Action>
   <ActionMatch MatchId="action-id">
    <ActionAttributeDesignator AttributeId="subject-id"
                               DataType="string"/>
     <AttributeValue DataType="string">view
     </AttributeValue>
   </ActionMatch>
  </Action></Actions>
 </Target>
 <Rule RuleId="R1" Effect="Permit">
 <Target>
  <Resources><Resource>
   <ResourceMatch MatchId="path-match">
    <AttributeValue DataType="pattern-path">
                                /Itinerary</AttributeValue>
    <ResourceAttributeDesignator
        DataType="simple-path" AttributeId="resource-id"/>
   </ResourceMatch>
  </Resource></Resources>
  </Target>
 </Rule>
</Policy>
```

Figure 5
Generated XACML policies during deployment

risks that may be identified through solution-monitoring activities. As we previously described, monitoring security situations in systems will likely cause changes in policies. These changes to security policies must be tightly controlled and access to them should be traced and audit trails supplied so that the processes may be adequately monitored.

IT administrators, security administrators, and operators play a fundamental part in managing and administering security policies for the enterprise, including those relevant to any specific solution.

As shown in Figure 4 (Policy Administration arrows), people playing these roles update security policies through the resource managers (e.g., application server runtime) or administer policies through security policy managers (e.g., Tivoli AccessManager).

Regardless of whether policy changes are made after initial subscription to a service or to reflect the state of the up-and-running services, the task of managing security policies is an important ongoing task throughout the lifetime of a solution.

Input

Input to the deploy-and-manage phase includes the output from the implementation phase and the binding of the solution to a given environment.

Output from the implementation phase includes security policies that are part of a packaged application, including those declared outside the application logic (targeted for the implementation or infrastructure) and those implemented as part of the application logic itself. In such cases, guidelines for declarative policies are typically part of the application package, because relating security concepts to implementation is better handled abstractly by an application developer or a solution integrator (who integrates applications to deliver a solution). For J2EE implementations, these initial policies are captured in deployment descriptors.

There are customizable parameters that bind a solution to a given environment. These parameters include initial configuration information that typically is not part of a solution package; these policies are communicated through out-of-band or necessary configuration information that "binds" a given solution to a given environment. For example, a user registry in a given enterprise may be a corporate LDAP directory, or it may be a custom-built registry. Other possible configuration and binding information includes trusted certificate authorities, key store, and certificate validation services that provide the initial topology to enable necessary isolation through demilitarized zone (DMZ) configuration, firewalls, and so forth.

Output

Given the lifetime of a solution and the importance of the deploy-and-manage phase, the output of that phase is not only about input that goes into the next phase (monitoring), but includes artifacts for the runtime entities in a system that must work with policies, including requestor runtime.

Conceptually, with every resource manager there is an associated policy manager. Access to a resource is controlled by the resource manager, which becomes the logical policy enforcement point (PEP) corresponding to that resource. This logical PEP performs access control by making decision requests to policy managers, which act as policy decision points (PDPs). Therefore, any policy changes, including initial policies at solution installation, refinement of policies during subscription, or changes to administered policies during management, are to be communicated between PEPs and PDPs. This is one of the output activities and part of the policy flow corresponding to this phase.

Another view of the policy flow is from the perspective of the requestor runtime. The client runtime may be required to be aware of certain constraints and requirements that are part of the solution policies. This is sometimes referred to as the "publish" task. For instance, the requirement to obtain access over a secure channel (e.g., SSL), or ensure confidentiality of a message (e.g., using message-level encryption) requires participation from both sides of the interaction, the requestor performing a task (e.g., encrypting a message) complemented by the action taken by the solution provider runtime (e.g., decrypting the message before processing).

Policy technologies

In order to meet the goals of coordinated policy enforcement and decision making, a consistent policy expression language is required for a given policy domain or functionality. For example, exchange of authorization policies between PDPs and PEPs is necessary, and given that multiple PEPs may consult a given PDP, the standard use of an expression language improves efficient implementation of policy expression evaluation and transformation adapters within the runtime. In the case of authorization policy, policy expression languages such as XACML help to articulate security policies consistently throughout the enterprise for different resource managers and solutions.

Distribution of policies between resource managers and policy managers must be standardized, and technology to do this must be made available. Note that in an SOA environment, resource managers and policy managers may be built and deployed as Web services in order to be accessed over defined protocols and bindings. After a policy distribution mechanism is available, it is used to provide automated distribution of policies between PDPs and PEPs. Distribution mechanisms may be used to combine the management of policies with the broadcasting of changes to appropriate PEPs or PDPs. In essence, distribution within a hierarchy of resource managers, as well as distribution between resource managers and policy managers, is required.

In cases where customization is necessary functionality for a consumer, template artifacts that capture security policies in a parameterized form, with the option of refining the policy information throughout the life cycle, are important. As described in the modeling section, policy templates may be defined in any of the upstream (early) stages, and the relation between these templates and the effective policies is handled by the underlying system. In the case of subscriptions, templates provide a mechanism to effectively customize the service instance for consumers, regardless of which presentation tool (e.g., a user interface tool, scripting tool) is used to present the options.

It may be necessary for policies to be distributed to the requestor runtime in order for the requestor to make necessary choices when submitting message requests. An expected approach is to use the WS-Policy[25,26] framework to publish Web Services policies, and for a requestor to use WS-MetadataExchange to retrieve relevant policies pertaining to a target service. In the case of security, WS-SecurityPolicy[27] framework would be used to express the security requirements and constraints as applicable to a requestor in order to access a given service.

Standards

Based on the various tasks and scenarios discussed in this section pertaining to the implementation phase, there is an identified requirement for consistency and standardization in various aspects of the policy life cycle in this phase in order to reduce the complexity. Some of the key areas where standardization is necessary are

- policy expressions to express domain-specific policies (e.g., WS-SecurityPolicy to express message security policies, XACML to express authorization policies),

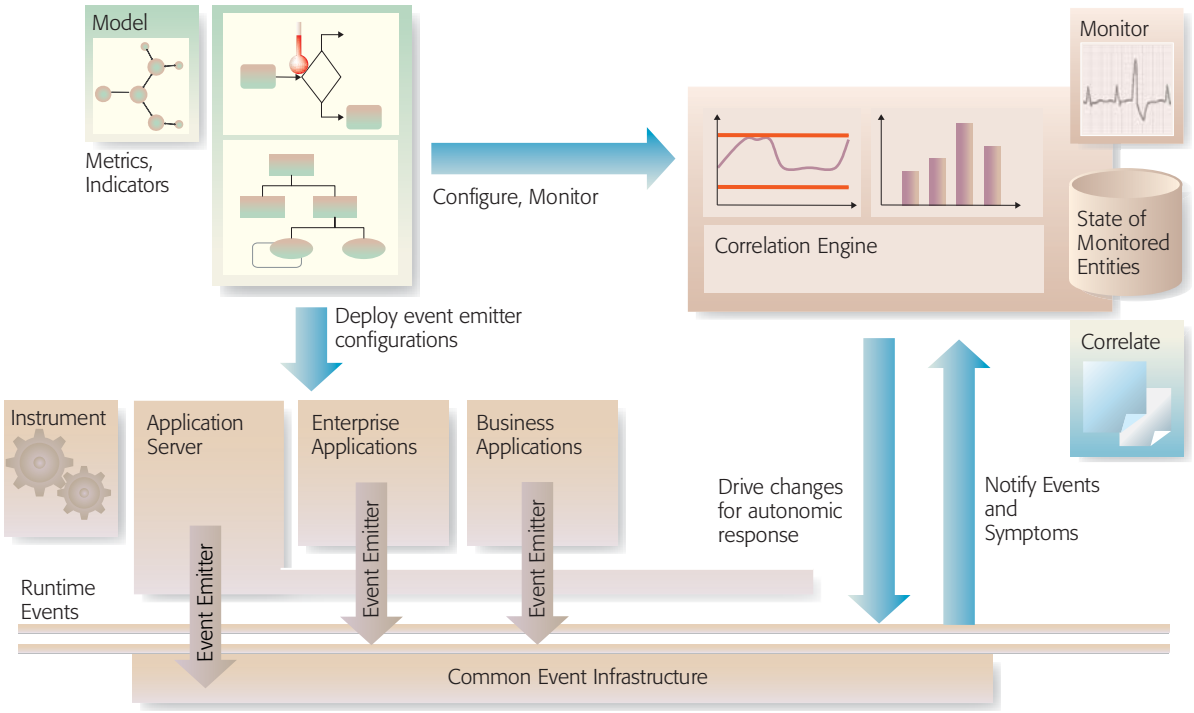

Figure 6
Flow of security events within a monitored system

- policy management services (e.g., WS-Authorization to manage authorization policies), and
- contracts between PEP and PDP proxies (local stubs specific to the bindings such as JACC in J2EE environments).

There is a requirement for a standard mechanism to publish requirements and constraints that a requestor must know in order to ensure interoperability between consumers and providers that are hosted on different platforms (e.g., WS-Policy framework, where the policies may be retrieved through means including WS-MetadataExchange).

MONITORING SECURITY SITUATIONS

Monitoring the health of enterprise systems and business applications may be performed by business stakeholders, IT operators, and administrators. Using a set of tools, consoles, and dashboards (e.g., Tivoli Enterprise Console*, and WBI Monitor), they monitor system behavior at runtime (e.g., denial-of-service attacks, or compliance with business security policies).

Based on changes which occur during the operation of the application, actions may be taken by administrators to manage or change policies or make changes to the deployment environment (e.g., take an application out of the network). Business analysts focus on business level monitoring (e.g., assess if key performance indicators are tracking business goals) while IT analysts focus on the underlying system events.

Solution behavior is monitored by observing actual runtime characteristics, through metrics monitored (e.g., number of authentication failures) in a given context and measured against the desired behavior. These desired metrics and the location of measurement points are derived through activities that occur during any of the phases—model, implement, or deploy. For example, as shown in ***Figure 6***, metrics, thresholds, limits, key risk indicators (KRIs), and KPIs may be defined at the business process modeling stage. These metrics are then transformed later in the life cycle into enforceable runtime metrics. Such business performance metrics may be correlated against a set of conditions (e.g., multiple authentication failures for a given user ID) to detect security situations (e.g., intrusion). Such security situations are of critical interest to the monitored solution behavior. These situations by themselves or

in a business context (e.g., a human-resource application being infiltrated) provide a sense of urgency for taking necessary action. Such actions may include alerting administration staff who may change policies as appropriate (as part of the management phase), or having the system autonomically react to such changes based on a set of predefined behavior rules (e.g., isolate set of applications from external access).

The monitoring phase of the solution life cycle consists of a few subtasks: monitor the measurement points, correlate and analyze security events, and plan actions. Actions may be planned based on IT information (e.g., a detected intrusion) or business impact (e.g., break in a trust relationship with a business partner). Thus system management tasks (e.g., isolate affected systems) or changes to business processes (e.g., change the trust relationship with a partner) may be appropriate actions.

Monitoring, analyzing, and planning tasks make up the entire monitoring phase of the solution life cycle. Details of the input, output, and technologies involved in this phase are discussed in following subsections.

Input

Similar to policy enforcement being application-managed or infrastructure-managed, instrumentation for certain monitored metrics may be either application instrumented or infrastructure instrumented. When policy enforcement is instrumented in the infrastructure, the monitoring infrastructure may be able to detect certain events. The input to the monitoring phase includes application-specific or application-independent monitored metrics, threshold values against which they are checked, and rules to help correlate events in order to detect security situations.

As depicted in Figure 6, during the model phase, the thresholds and limits are modeled for desired performance indicators or metrics to be monitored. During the implementation phase, these metrics are either instrumented in the application, or declared outside the application for the infrastructure to monitor the metrics. Based on the behavior against the metrics, events are generated by the resource managers, including applications. These events include security events. When these events flow over a common event infrastructure, they may be correlated by event-filtering engines.

Output

Using a monitor framework and dashboards, these events and situations may be compared against monitoring thresholds. Output from the monitoring phase and the monitoring systems is a set of: (1) detected situations, including deviations in metrics or anomalies (e.g., abnormal credit card activity), (2) alerts for appropriate action, and (3) possible relevant actions that may be recommended for the events analyzed and their estimated impact. This is illustrated in the top right part of Figure 6. These alerts and information may be sent back to the individuals acting in the appropriate roles (e.g., security administrator, IT administrator, business analyst, etc.). In an automated system, one such output may be another higher-level event that helps to capture the impact of certain low-level events.

Policy technologies

A common event infrastructure is necessary to support various events (IT and business), propagate them across various components in a given system, and allow for rules and plug-in points to perform filtering, analysis, and correlation of events. These may be based on a message infrastructure. In order to efficiently filter, correlate, and thus handle multiple events, a common event format may be used to capture event information consistently (e.g., using Common Base Event format [CBE]).[28]

In addition to support to allow instrumentation and to flow and filter events, an ability to define rules for event correlation is necessary. Visualization of event situations helps depict the impact of a situation to both the IT and business environments. Event dashboards and relevant action management consoles may be used to define recommended actions.

Standards

Due to the nature of various systems from various vendors that are integrated in an enterprise, it is important to have a standard mechanism such as CBE to create events in a common format and then send them over a common event infrastructure. This is required for interoperability among systems.

SUMMARY

We have outlined a policy-driven approach based upon business and model-driven development and

management methodologies to achieve effective management of security policies for applications. This approach helps meet the changing requirements of an on demand business. It has factored in the interactions among the different people playing different roles in an organization and the importance of tools to help them perform their responsibilities in a consistent manner. We described the tools, technologies, standards, and runtime necessary to meet the requirements of managing security policies during the life cycle of a business application. This proposal used a pragmatic approach to find intersection points between platform-independent modeling of security policies and the concrete articulation of policies and their enforcement. This type of approach offers a way to leverage the monitoring of adherence and compliance to policies in both IT and business dashboards and to manage and map the relationship between business artifacts and implementation artifacts, so that business policies are reflected in implementation. We outlined a set of technologies and tools that should be provided. We described runtime enhancements and a dashboard to help monitor the security policies throughout the life cycle. We recognized the importance of open standards to enable an on demand business and proposed areas where extensions and enhancements in standards should be introduced.

ACKNOWLEDGMENTS

The authors would like to acknowledge the input from various people in IBM who have directly or indirectly contributed towards some of the concepts identified in this paper. Special thanks go to the following colleagues: Donald Ferguson, John Sweitzer, Sridhar Iyengar, Jeffrey Frey, Michael Swanson, Mark Linehan, Allen Gilbert, David Kaminsky, Asit Dan, John Rofrano, Balaji Krishnamachari, Clare Marie Karat, John Karat, Carolyn Brodie, Steve Adler, Rick Cohen, and Jayashree Ramanathan.

CITED REFERENCES

1. *IBM Rational Software Development Platform*, IBM Corporation, http://www-128.ibm.com/developerworks/platform/.
2. OMG Architecture Board MDA Drafting Team, "Model Driven Architecture: A Technical Perspective," Object Management Group, http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01.
3. *Model Driven Architecture*, Object Management Group, http://www.omg.org/mda.
4. G. Booch, A. W. Brown, S. Iyengar, J. Rumbaugh, and B. Selic, "An MDA Manifesto," Chapter 11, *MDA Journal*, D. S. Frankel and J. Parodi, editors, Meghan-Kiffer Press, Tampa, FL (2004).
5. C. Alberts and and A. Dorofee, *Managing Information Security Risks: The OCTAVE (SM) Approach*, Addison Wesley Professional, Boston, MA, ISBN: 0-321-11886-3 (2003).
6. *FISMA Implementation Project*, National Institute of Standards and Technology, http://csrc.nist.gov/sec-cert/.
7. *Business Process Execution Language for Web Services*, BEA Systems, IBM, Microsoft, SAP AG and Siebel Systems, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/.
8. Sarbanes-Oxley Act of 2002, http://www.sec.gov/about/laws/soa2002.pdf.
9. *Business Process Modeling Notation Information*, Business Process Management Initiative, http://www.bpmn.org/.
10. *Business Process Definition Meta-model*, http://www.omg.org/cgi-bin/doc?bei/2003-01-03.
11. *Unified Modeling Language*, Object Management Group, http://www.omg.org/uml.
12. *Eclipse Modeling Framework*, Eclipse Project, www.eclipse.org/emf.
13. M. Pistoia, N. Nagaratnam, L. Koved, and A. Nadalin, *Enterprise Java Security: Building Secure J2EE Applications*, Addison-Wesley Professional, Boston, MA, ISBN: 0-321-11889-8 (2004).
14. L. Koved, A. Nadalin, N. Nagaratnam, M. Pistoia, and T. Shrader, "Security Challenges for Enterprise Java in an e-business Environment," *IBM Systems Journal*, **40**, No. 1, 130–154 (2001).
15. S. Johnston, "Modeling security concerns in service-oriented architectures," IBM developerWorks, http://www-106.ibm.com/developerworks/rational/library/4860.html?ca=dnp-322.
16. J. Asensio, V. Villagra, J. Lopez, and J. Berrocal, "UML Profiles for the Specification and Instrumentation of QoS Management Information in Distributed Object-Based Applications," *Proceedings of the Fifth World Multi-Conference on Systemics*, Cybernetics and Informatics, ISBN: 980-07-7543-9 (July 2001), pp. 22–25.
17. T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," *Proceedings of the 5th International Conference on The Unified Modeling Language*, pp. 426–441 (2002).
18. *Java 2 Platform, Enterprise Edition (J2EE)*, Sun Microsystems, http://java.sun.com/j2ee/.
19. M. Hondo, N. Nagaratnam, and A. Nadalin, "Securing Web Services," *IBM Systems Journal*, **41**, No. 2, 228–241 (2002).
20. S. Johnston, "UML Profile for Software Services," IBM developerWorks, http://www-128.ibm.com/developerworks/rational/library/05/419_soa/.
21. P. Kruchten, "The Rational Unified Process: An Introduction (2nd Edition)," *Addison-Wesley Professional, Boston, MA* (2000), ISBN: 0-201-70710-1.

22. *Java Authentication and Authorization Service API*, Sun Microsystems, http://java.sun.com/products/jaas/.
23. *Java Authorization Contract for Containers API*, Sun Microsystems, http://java.sun.com/j2ee/javaacc/.
24. *OASIS eXtensible Access Control Markup Language*, OASIS, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.
25. *Web Services Policy Framework*, IBM, BEA, Microsoft, SAP, Sonic Software, Verisign, http://www-128.ibm.com/developerworks/library/specification/ws-polfram/.
26. *Security in a Web Services World: A Proposed Architecture and Roadmap*, IBM and Microsoft, http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/.
27. *Web Services Security Policy*, http://www-128.ibm.com/developerworks/library/ws-secpol/.
28. *Common Base Event Specification*, IBM Corporation, http://www-128.ibm.com/developerworks/webservices/library/ws-cbe/.

Accepted for publication June 16, 2005.
Published online October 25, 2005.

Nataraj Nagaratnam
IBM Software Group, 3901 S. Miami Blvd, Durham NC 27703 (natarajn@us.ibm.com). Dr. Nagaratnam is the Chief Architect for Identity Management and lead security architect for on demand security infrastructure and technical strategy. As a Senior Technical Staff Member, he drives security architecture and design activities across IBM products and platforms. In his career at IBM, he has been the lead security architect for WebSphere Application Server and then, the lead security architect for the WebSphere Platform. He leads and participates in various open standards activities in standards organizations, including JCP, OASIS, WS-I, and GGF. He has authored and co-authored numerous journal papers, books, and security specifications, including *Enterprise Java Security* published by Addison Wesley.

Anthony Nadalin
IBM Software Group, 11501 Burnet Road, Austin TX 78758 (drsecure@us.ibm.com). Mr. Nadalin is the chief security architect for IBM Software Group. As a Distinguished Engineer, he is responsible for security infrastructure design and development. He serves as the primary security liaison to Sun Microsystems JavaSoft Division for Java security design and development collaboration. In his 22-year career with IBM, he has held the following positions, lead security architect for VM/SP, security architect for AS/400, and security architect for OS/2. He has also authored and co-authored over 40 technical-journal and conference articles and published several books on Java security and the Internet.

Maryann Hondo
IBM Software Group, One Rogers St, Cambridge MA 02142 (mhondo@us.ibm.com). Ms. Hondo is the Web Services Security Standards lead in Emerging Technology for IBM Software Group. She joined IBM/Lotus in 1996 as security architect for the Lotus e-Suite, participating in the development of Java security (JAAS). Her previous background includes working for Hewlett Packard Corporation on DCE and PKI Smartcard-based Single SignOn, working for Digital Equipment Corporation on a B1/CMW operating system, and working for AT&T Bell Labs on B2 Unix. She is one of the co-authors of the WS-Security, Policy, Trust, and Secure Conversation specifications announced by IBM and other business partners in 2002–2004. Before joining the Emerging Technology group, she managed the IBM/Tivoli IETF PKIX reference implementation development group (Jonah) and was part of a research team working on service-oriented architecture, which produced the first Emerging Technology toolkit (see IBM developerWorks™). Her standards activities include chairing the working group on Single Sign On at the Open Group, chairing the Security team for ebXML, leading the Security working group at UDDI, participating in OASIS technical committees (SAML, WS-Sec, and XACML), and participating in the Open Mobile Alliance, MWS, and Security working groups.

Michael McIntosh
IBM Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne N.Y. 10532 (mikemci@us.ibm.com). Mr. McIntosh is a Senior Software Engineer in the Java and Web Service Security Group at the Watson Research Center. He represents IBM in the WS-I Basic Security Profile working group as an editor of the Profile and in the OASIS Web Services Security technical committee. He has worked in the information technology industry for 22 years and for the past three years at IBM.

Paula Austel
IBM Thomas J. Watson Research Center, 19 Skyline Drive, Hawthorne, N.Y. 10532 (pka@us.ibm.com). Ms. Austel is a Senior Software Engineer in the Java and Web Service Security Group at the Watson Research Center. She has participated in the following: OASIS Web Services Security technical committee, OASIS Security Services technical committee, and WS-I Basic Security Profile working group. ■