A framework for eGovernance solutions

P. A. Mittal
M. Kumar
M. K. Mohania
M. Nair
N. Batra
P. Roy
A. Saronwala
L. Yagnik

This paper presents a framework which simplifies the task of developing, deploying, and managing complex, integrated, and standards-compliant eGovernance solutions. The framework enables development, configuration, integration, and management of solutions at a higher semantic level. It also provides commonly used services such as access to citizen and property records, access control and authentication services, public key infrastructure, and support for digital signatures. The ability to manage solutions at a higher semantic level enables administrators who are not proficient in programming to customize solutions in order to address specific needs of the different national, state, and local governments. This includes the ability to customize interfaces for multiple local languages used in government transactions and to customize workflows to conform to the organizational structure and policies to manage access to and retention of government records.

1. Introduction

The term *eGovernance* refers to the process of using information technology for automating both the internal operations of the government and its external interactions with citizens and other businesses. Automation of internal operations reduces their cost and improves their response time while at the same time allowing government processes to be more elaborate in order to increase their effectiveness. Automation of interactions with citizens reduces the overhead for both the government and the citizens, thus creating value for the economy. As an example, consider an online service that can be provided by the transport department for the renewal of driving licenses, currently a leading eGovernance application in India. At present, the application works as follows: The applicant visits the regional transport office, completes the renewal form on paper, and submits the form to a clerk, along with a photograph, proof of residence, proof of date of birth, and transaction fee. The clerk processes the application form manually. The applicant typically has to wait in the office for several hours before receiving the renewed driving license. Besides the inconvenience to the applicant, previous traffic violations are not properly verified, and there is no provision for easy management of expired license records.

With the deployment of the eGovernance framework, we expect the following improvements. In a typical scenario, persons visiting the state government portal can choose to renew their driving licenses by completing the renewal forms online. In the future, the information could be digitally signed by the citizen to ensure nonrepudiation using the public key infrastructure of the eGovernance framework, possibly managed by the government. The solution verifies the applicant's digital signature, residential address, and traffic violation records in real time using the citizen records maintained in the framework and support for inter-agency collaboration. It then requests that the applicant make online payment for the renewal fees by means of credit card, debit card, etc. On verifying the payment details with a payment server, the application is added to the list of driving license applications to be approved, and a notification is sent to the approving authority in the government. The approving authority logs on to the state portal, views the pending renewal applications, and approves or rejects them. Approved applications are automatically forwarded to the license printing application. The driving license card is then sent to the applicant by courier. The system periodically archives or purges the renewal applications and archives the expired driving license records. The

0018-8646/04/$5.00 © 2004 IBM

accepted and rejected applications may have to be purged on different schedules.

The design and development of such complex solutions[1] poses significant challenges. One such challenge is that in current development environments [1–4], the application developers have to work at a low level of abstraction. This means taking care of low-level issues such as interprocess messaging, tools integration, and data modeling while defining the application logic. Similarly, solution reconfiguration and management requires the solution administrator to have a detailed understanding of the application logic, making the task time-consuming and error-prone. Handling these challenges effectively requires highly skilled and experienced information technology (IT) professionals, increasing development costs for effective eGovernance solutions. Solution administrators typically lack these IT skills, rendering change management impossible.

In solutions developed to date [5–8], each eGovernance solution has customized existing products to address an individual government agency requirement. However, this might not always be the most economical way to develop a solution. In most industries, around 85% of the processes are same across companies within that industry. A similar fraction of the processes can be expected to be similar across different government solutions. Clearly, it is desirable to develop these processes once and then reuse them for many solutions. This is also likely to be true for data models, user interfaces, etc. For example, the address verification process in the driving license renewal solution considered above can be reused while developing a passport renewal solution. Similarly, the traffic violation record verification process can be offered as a service to insurance businesses to be reused in a car insurance solution. Lack of information (metadata) on available processes and components and difficulty in customizing these for a specific need currently hinder their reuse for multiple solutions.

One can readily conclude from the preceding discussion that there is a need for a framework that can simplify the development, deployment, and management of eGovernance solutions. The eGovernance framework proposed in this paper addresses the requirements identified in the preceding discussion by

- Enabling modeling of a hierarchy of building blocks that can be used to abstract government process to a higher semantic level.
- Enabling specification of workflow for government processes independent of standards; the platform takes care of generating the deployable solution that conforms to the appropriate standards.
- Enabling reuse of effort across solutions by providing tools to develop generic, parameterized applications or processes that can be stored in a repository with appropriate metadata and effectively reused by various applications with appropriate customization.
- Extending programming models to specify the customization points in an application or solution during development, and intuitive interfaces to enable modification of solutions easily after deployment without the need for the business user to modify the application source code.
- Extending programming models to simplify specification of multilingual and multidevice interfaces.
- Providing tools to author the wrappers for the legacy applications and workflows integrating multiple applications to automate processes spanning several government agencies.

The current framework prototype described in this paper will evolve with customer engagements. The approach is to maximize the reuse of available tools and middleware. In the initial stages, the focus of the effort is on the Indian eGovernment market; however, the platform can be extended to suit the needs of other countries as well.

The rest of the paper is organized as follows. Section 2 presents the key requirements of eGovernance solutions. Section 3 describes a model for eGovernance solutions and discusses the framework components in relation to the solution model. Section 4 describes how the eGovernance framework simplifies solution development, deployment, and management. The eGovernance framework architecture and the detailed description of its components appear in Section 5. Section 6 explains the framework with sample scenarios and user interfaces. Section 7 discusses the prior work in this area, and Section 8 presents the conclusions and future directions.

2. Requirements of an eGovernance solution

To understand how the eGovernance framework simplifies the task of developing eGovernance solutions, it is instructive to acquire a perspective on what constitutes an eGovernance solution and the aspects in which eGovernance solutions differ from enterprise solutions. Over the last few years, eGovernance has been implemented as isolated applications for specific government departments, creating a complex web of interactions among departments that deliver similar services to customers. Counter to this trend, today's need is for government to present a single face to citizens. To address these forces, government must transform its isolated, fragmented framework into a collaborative,

[1] In this paper we use the term *application* to refer to a programming asset developed by a vendor under one contract. The term *solution* is used to refer to a collection of applications, possibly deployed on heterogeneous platforms, that are integrated to enable the delivery of a government function.

externally focused, segment-centric operating model. The segment approach reorganizes services from a holistic understanding of customer needs that span traditional organizational boundaries. This in turn provides to the citizen an integrated view based on a specific eGovernment function. Some representative requirements of typical eGovernance solutions are presented below.

- *Security and privacy* Authentication requirements for government solutions are significantly more stringent than those for typical business applications. The dealings of an individual with a business are less common than those of an individual with a government, since most of us file taxes and apply for driving licenses. While businesses can limit their liability by requiring payments before goods are shipped, government cannot afford to deal with imposters with profit motives or malicious intent. Identity theft is only one of the problems that can be caused by poor authentication in eGovernance applications. The sheer volume of information that governments maintain on individuals makes privacy issues extremely important. Since all eGovernance applications require strong authentication processes, possibly involving multiple government agencies, it is imperative to offer authentication as a government-managed service in the eGovernance infrastructure, as proposed in this paper. The government, or its delegate agency, must manage the public key infrastructure needed for this purpose.
- *Electronic receipts and payments* A large number of government applications require the ability for users (citizen or business) to make and receive payments for services received from the government, often maintenance of or information on records such as property deeds. Electronic forms of payment are not yet popular in developing countries. Because governments, compared with businesses, have fewer methods for collection from delinquent payees, robust payment methods are essential services for the eGovernance framework.

  This is also an area in which IT can dramatically increase government efficiency. In an ongoing study of actual government processes, it has been observed that each government agency has its own departments for collecting payment. With proper business reengineering and workflow integration capabilities, unified payment collection can be offered as a service to all government agencies.
- *Record management* Government processes deal with a large number of records managed by different departments. The records have different contexts; some are more permanent in nature than others. For example, felony convictions are permanent records, while traffic misdemeanors must be deleted or archived after a certain time duration. The eGovernance framework must provide a record management system to electronically capture, preserve, manage, protect, and ultimately dispose of records. Proper audit trails are also important. The system must be as automated as possible to minimize the involvement of skilled database administrators; at the same time, it must be customizable to the needs of different agencies and different record types. Tools and services for record management will be the most widely used services of the eGovernance framework. (Record-keeping by governments or temples may be credited with driving the invention of writing itself, albeit on clay tablets, in Sumerian times.) Compared with developed countries, relaxed accountability standards for government employees in developing countries make record management and the preceding requirements essential for the eGovernance framework.
- *Intuitive graphical/conversational interfaces* Literacy in developing countries is significantly lower than that in developed countries. The eGovernance applications should be as accessible to literacy-challenged users as to literate ones. Furthermore, even literate users may require their native vernacular instead of English as a medium of communication. This requires the applications to communicate with the user in the user's preferred language, by means of intuitive graphical and conversational interfaces. For example, the denomination of an Indian currency note is written in 15 different regional languages on the currency bill itself!

  The penetration and adoption rate of cell phones compared with PCs in developing countries is much higher than that in developed countries. From the perspective of a developing country, the eGovernance application must be able to provide consistent interaction for users regardless of the type of access device—laptop, desktop, personal digital assistant, (PDA), or cell phone. The important issue related to supporting interactions through such mobile devices is that the applications have to deal gracefully with occasional disconnects from the network.
- *Record virtualization* Typically, government departments contain multiple subdepartments distributed across geographical locations. The eGovernance IT infrastructure, comprising browser-based clients, Web servers, application servers, and database servers, must be deployed in a distributed manner. Location of the database servers is perhaps the most critical decision, because the records being maintained by various agencies are likely to be required by applications of other agencies. This has been identified as a critical requirement in multiple ongoing customer dialogues, described later in the section on customer engagement experience. High availability of these records is critical

to the smooth operation of the eGovernance infrastructure. Record location can be made transparent to the applications by making it available as a Web service and encoding it in XML-based standards. This also enables applications to access heterogenous data from disparate and distributed sources without being aware of or concerned about the complexity involved in accessing the data.

- *Service integration* Government departments must communicate and interoperate with their suboffices in remote areas of the country as well as with other departments and businesses. This requires the ability to connect to pre-packaged applications as well as to the legacy applications developed by different vendors, involving both application connectivity and process integration. Currently, in the absence of any eGovernance standards or reference architecture across the country, solution vendors are developing government applications in an *ad hoc* manner, on various platforms, using different programming languages and runtime environments. The eGovernance framework prescribes open-standards-based interoperability guidelines, analogous to the eGovernment Interoperability framework [9], to allow solutions to be created on the basis of reusable building blocks that span multiple government departments.
- *Solution customizability* In the eGovernance environment, government processes must be modified periodically to reflect changes in organization, policies, work realignment, etc. For example, the target group characteristics for a social welfare scheme may change over a period of time, or the policy to grant approval of the layout of a new building must be modified in the future. It is important that the developed applications should be flexible enough to allow such changes, even after deployment. Moreover, this reconfiguration process must typically be carried out by officials not skilled in IT. Thus, the eGovernance application should provide an intuitive user interface to define, manage, and modify these processes. As a related issue, an eGovernance application deployed across the country has to be aware of the local culture, local government policies and procedures, etc. The application should be customizable along these dimensions as well.

The representative requirements presented above are not specific to a particular application or country. While these requirements arise in enterprise (commercial) applications as well, they become more significant in eGovernance applications for a variety of reasons—the sheer scale (for example, India is a country with more than a billion people, with only a small fraction on the privileged side of the "digital divide," and usability is a major issue); the higher stakes involved in security and privacy; the lack of reliable infrastructure (reliable connectivity, for example, could be a major issue in remote rural areas); and, most significantly, the budgetary constraints. While a development platform that reduces the complexity of addressing one or more of the above requirements benefits application development in general, it will have a much stronger impact on the development of eGovernance applications in particular. How this is enabled by the proposed eGovernance framework is discussed in the next section.

3. eGovernance solution components and the eGovernance framework

In this section, we describe the common components of various eGovernance solutions and discuss various IT technologies that comprise the eGovernance framework [10]. We then discuss how the development, deployment, and management of the various components of the eGovernance solutions are supported by the technologies of the eGovernance framework. Finally, we discuss how the framework helps in integrating the solution components into customizable processes.

The eGovernance solution components can be organized in three tiers, as shown in **Table 1**, with the first tier representing the information services that use Internet technologies to provide access to information. A range of solutions can be built based on access to public information alone. With proper authentication, in order to protect privacy, access can be extended to personal information for individuals. The next tier of solution components add transactional capabilities to the services, enabling citizens to initiate and interact with various government processes. Components in this tier also enable employees of a government agency or department to initiate/interact with processes of other government departments. The third tier (data mining and analysis) represents the strategic future of eGovernance solutions, which will enable the on-demand vision of a government that is responsive to social, economic, and political changes and resilient to these changes because of the flexibility provided by the new capabilities of eGovernance. eGovernance will enable governments to extract trends from the transactional data, make meaningful forecasts from these trends, and implement the necessary policy decisions rapidly.

The eGovernance framework, like frameworks for many industries, can be viewed as comprising three technology layers: the *enablers or tools* for development of eGovernance solution components, the *middleware or framework services* for runtime support of eGovernance applications or solution components, and the *solution environment,* which helps in creating effective inter-agency processes from the solution components. These three layers comprise the technology view of an eGovernance

Table 1 Model for the eGovernance solution.

	Revenue	*Law and order*	*Health and basic education*	*Public works*	*Planning controls (licensing and permits)*
Information services · Public information · Personal data	· Instructions for compliance	· Police advisories · Interaction with judiciary	· Educational material	· Postings for RFP/RFQs (tenders) · Contract awards	· Rules and regulations · File tracking
Transactional services	· Payments for permits, licenses · Tax filings	· Online complaints · Tracking progress in investigation	· Patient health records · Appointments for public hospitals	· Procurement contracts	· Driving license renewal · Birth and death registration
Data mining and analysis	· Regional compliance trends and enforcement measures	· Trends in crime and violations; deployment of resources	· Early warning of epidemics; deployment of control measures	· Fraud detection in contract awarding and procurement	· Population growth forecast and related control measure

Table 2 eGovernance framework: Solution components and technology.

	Information services	*Transactional services*	*Data mining and analysis*
Solution environment	· Single, unified portal for information access to citizens, businesses and employees	· Messaging between heterogeneous systems · Collaboration across various departments · Workflow spanning multiple agencies	· Change management
Services	· Multilingual content management · Authentication and privacy · Multidevice access management	· Data models and government record services · Commonly used process patterns · Payment services	· Data integration · Data analysis
Tools	· Content authoring and management · User role and account creation · Multiple device authoring	· Record management policies specification · Business object creation · Legacy wrapper development	· Data attribute mapping for record virtualization

solution. The eGovernance framework proposed in this paper is expected to simplify the development, deployment, and management of eGovernance solutions. While solutions are currently developed as isolated applications for specific departments, the goal is to develop applications and solution components that can be used by multiple organizations and that are designed to share information with related components, becoming part of an integrated solution. As illustrated in **Table 2** and discussed in detail below, the enablers (tools), middleware (services), and solution environment provide support for the eGovernance solution components in all three tiers of Table 1.

Enablers, or tools

The enablers are the tools used to develop the high-level reusable building blocks of an eGovernance solution. The building blocks become part of the framework services layer. They provide eGovernance application-specific semantics, allowing application development at a higher

semantic level. The low-level details are incorporated by the development platform as it generates the code. The building blocks can be customized to address the requirements of the application being developed through *wizards.* (A wizard is software that guides the developer, through a set of screens, to easily customize a solution component for a specific application.) The following enablers address the representative requirements listed in the previous section:

- *Security and privacy enabler* This helps create the components responsible for user account maintenance, user authentication (log-in, password management), as well as access control. The wizard available with this enabler customizes associated policies such as password policies and access control policies. It relies on the public key infrastructure of the framework services.
- *Electronic receipts and payments enabler* This helps build the application component responsible for supporting the financial transactions for multiple government agencies. Again, the accompanying wizard can be used to specify the entities involved (i.e., government agencies and the financial institutions taking care of the transactions) and customize the agency-specific payment policies.
- *Record management enabler* This helps in specifying the policies for management of transactional records at a higher level in terms of business objects and in tying these policies to the relevant information sources. The policies may pertain to retention (archival/deletion schedule) for records, audit trails for changes, encryption policies for storage and transmittal, logging of requests, and acknowledgment of successful transmission.
- *Interface enabler* This helps build the citizen-facing graphical and multilingual conversational interfaces for the application using reusable user interface components and technologies for language translation, speech-to-text, and text-to-speech features. The wizard also allows the developer to author the user interface in a device-independent manner and then select the client devices that should be able to access the resulting application.
- *Record virtualization and data integration enabler* This helps to provide a virtual view of the government records for the eGovernance applications. The accompanying wizard allows the developer to specify the named data sources or virtual records and their schema/attribute mappings, etc., while their actual location is specified at the application deployment time.

In addition to the prepackaged enablers, the eGovernance framework can be extended with new enablers and the accompanying configuration wizards. Development organizations can create their own enablers or extend existing enablers for their own use, thereby creating a hierarchy of enablers.

Middleware framework services and reusable assets

The framework services layer provides the domain-specific services and data models that can be used in multiple government solutions, possibly across different departments. For example, a large number of government applications refer to common entities such as citizen, land, and establishment. The data models and corresponding records for these entities can be reused, not only saving development time and effort but also ensuring consistency of data and ease of sharing it across applications. Access to the citizen database, land records database, etc. can be supported using Web-services-based interfaces. Government applications also have similar processes, such as user identification and authentication, address verification, medical certificate verification, audit, and logging. These processes can be provided as the framework services to be used across multiple government solutions. These services use the high-level reusable building blocks, created using the enablers in the preceding section, such as access control policies, payment policies, record management policies, business objects, and multilingual interfaces.

The generic forms of frequently used applications or solution components themselves can be offered as Web services from the eGovernance middleware. The driving license renewal and the passport renewal applications are examples of application templates that can be reconfigured and reused with customizations in different solutions. Each solution implemented using customized application templates may differ in its look and feel or some of the processes, but the core application logic does not change. This addresses the important issue of reusability of programming assets by vendors developing government applications, who need to develop similar applications, with minor variations, for various state governments, across multiple countries.

The eGovernance framework will initially incorporate frequently used assets such as the common data entities, their use-case diagrams, commonly used processes and services, and application templates. Furthermore, the framework will provide the capability for the developers and the domain experts to create, categorize, and store new assets, the details for which are discussed in the section on framework architecture.

The commonality among the eGovernance applications and solution components may not be limited to the common data models and the services described above. Every industry has a set of processes that implement the typical services provided by the industry. Business experts abstract these into identifiable industry patterns. Examples

of some common patterns observed in government processes are lodgement with and without payment, auditing, procurement and tendering (RFQ), and public complaint. For instance, the lodgement without payment pattern enables the citizens to submit and/or register documents for various government services. Examples of government services that can be developed using this pattern are registration of deeds, companies, dealerships, licenses and permits; applications for caste certificates, nativity certificates, and birth certificates; and filing of tax returns. The eGovernance framework provides the ability to use the eGovernance patterns in facilitating the development and deployment of government applications. It also provides a few predefined patterns encountered frequently in government application requirements.

Solution environment

The solution environment provides developers and solution managers with the capabilities necessary to create and manage solutions spanning processes in multiple government agencies by integrating the applications, solution components, and processes across these agencies. This involves defining new services, integrating legacy services by developing wrappers around them, and messaging between applications. These capabilities are built on top of existing integration solutions such as the WebSphere* Business Integrator [11]. The WebSphere Business Integrator delivers five key integration capabilities: Model, Integrate, Connect, Monitor, and Manage. These capabilities are supported by a common framework consisting of tooling, business objects, and adapter framework, a services-oriented architecture, and a browser-based graphical user interface (GUI). Further, multiple templates can be created using the WBI to handle process variations in different states. The accompanying wizard hides much of the complexity involved in this customization and also ensures that the above tasks are done in a standards-compliant manner.

In addition to the integration capabilities, effective solutions also require the involvement of government employees in the eGovernance processes. The framework will employ IBM portal technologies [12] to bring employees of different relevant agencies, information from independent government record repositories, and portlets for different government applications in one Web-browser-based view to enable involvement of employees from various government agencies in the solutions spanning them. The portal server provides quick integration with back-end systems to build portlets for integration with relational databases, domino databases, and enterprise applications (SAP**, Siebel**), etc. The eGovernment portal can be further integrated with content management tools such as the DB2* Content Manager [13]. This tool can integrate and leverage all forms of content—document, Web, image, rich media—across diverse business processes and applications, including Siebel, PeopleSoft**, and SAP. It also helps in Web content development without the need for programming skills.

4. Simplifying eGovernance solution development, deployment, and management

In this section, we describe how the eGovernance framework simplifies solution development, deployment, and management. This addresses the ways in which a developer can create a new solution, an administrator can configure and deploy it in different environments, and a business user can manage it after deployment.

Solution development

Let us first see how the framework helps a software developer in the solution development process. The framework provides tools and services to enable application modeling at a higher, semantic level. The solution components encapsulate the industry standards and best practices, leading to the development of high-quality, interoperable applications. Moreover, the framework implementation is based on the open extensible Eclipse [14, 15] platform, which provides an integrated development environment with a uniform look and feel. This allows the developers to seamlessly use multiple application development tools from various vendors. Additional details are given later in the paper.

Furthermore, the framework reduces the solution development time by providing the capability to reuse assets across solutions developed by multiple vendors. Specifically, it allows the developer to package a set of related artifacts as an asset and store the asset in a repository tagged with a set of keywords, description and, if needed, the metadata to facilitate its reuse. The metadata may include the configurable parameters of the asset. While developing an application, another developer can search the repository for available assets based on the keywords. For each asset returned, the developer can read its description and may decide to use it in the application. The framework provides a wizard that reads in the asset metadata and assists the developer in using the asset in the given application.

Figure 1 shows how multiple developers (independent software vendors, or ISVs), each developing their own independent eGovernance applications, share assets among themselves through a central repository of assets. It is clear that good specifications by architects of eGovernance applications call for the application developers to package all potential assets as reusable components. This repository can be owned by the government and can be hosted and managed by the

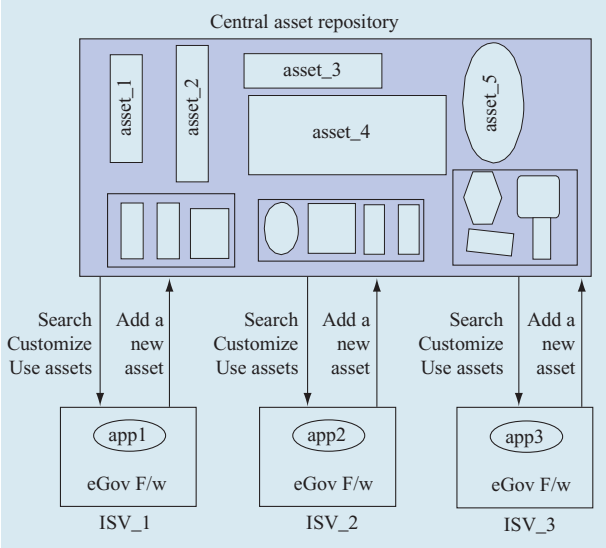

Figure 1

Multiple ISVs reusing and sharing assets.

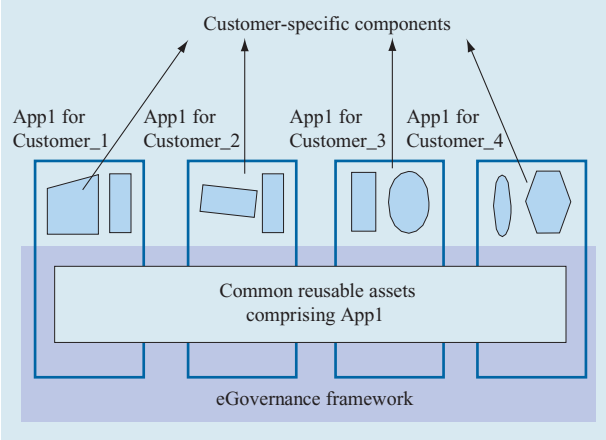

Figure 2

Single ISV customizing an application for different customers.

government or by an independent third party. **Figure 2** shows how a particular developer reuses a given asset in the repository for related requirements across multiple government customers.

Solution deployment

This section describes how the framework simplifies the task of deploying a solution. As discussed earlier, an eGovernance application typically has to be integrated with a diverse set of other applications to create a solution. To keep the deployment complexity to a minimum, it is important that the application being developed adhere to certain standards—DCA21 standards, standards for Web services, data exchange, business processes, messaging and connectivity standards, etc. This allows for easy integration with other components and applications in the deployment environment. For instance, a Java 2 Platform Enterprise Edition (J2EE) [16, 17] application can be packaged as a set of WAR and EAR files that can easily be deployed on any J2EE-compliant application server. Developing the application at a higher semantic level using the enablers has the advantage that standards compliance is now the onus of the enabler; the generated modules are guaranteed to be standards-compliant, and, as a result, easily deployable. Consider the renewal of driving license example described in Section 1. The driving license solution may comprise multiple, distributed component applications. For instance, there may be one standalone application deployed by the state police department to handle verification of traffic violations, another by a municipal corporation to handle address verifications, and yet another deployed by a participating financial institution to handle online payments. These applications, if developed on the basis of the proposed framework, will be Web services that can integrate easily across multiple departments, using the BPEL4WS [18] standard. In another scenario, if the online payments application already exists on an enterprise information system, the solution environment of the framework can use the J2EE Connector (J2C) connectivity standard [19] to wrap the legacy application. The wrapped legacy application can then be invoked like any other Web service. Since the interface between the driving license application and the financial institution application is standards-compliant, the transport department has the liberty to replace one financial institution with another with no disruption in service. In yet another scenario, if the J2EE-compliant [16, 17] driving license application has to communicate with the police department legacy application using WebSphere Message Queue [12], the framework solution environment supports the communication based on JMS [20, 21].

In addition to adherence to standards, there are building blocks that ease solution deployment in different environments such as access from different devices, specification of captions in multiple languages, and access to data from disparate and distributed data sources by simplifying the associated configuration and attribute mappings. Also, the administrator can easily define the security policies or record management policies according to the needs of the specific customer. Thus, the framework assists an administrator in solution deployment.

Solution management

Next, let us look at the complexity of managing the eGovernance solutions. The need to make changes in solutions arises from changes in the policies, laws, business processes, and even standards as and when they are upgraded. For example, the government might stipulate at a later time that the driving license be renewed only if the applicant has a clean tax record. This verification requirement adds more fields to the forms [for example, the Personal Account Number (PAN), which is similar to the Social Security Number in the U.S.] and requires the taxation department application to be added to the workflow as well. It is important that government applications be flexible enough to incorporate these changes without substantial reprogramming effort and with as little downtime as possible. Change management in applications developed using the framework requires little effort, since the building blocks allow the reuse of earlier development effort to the maximal extent possible. The developer changes the application process at the semantic level and modifies the associated building blocks. The framework then regenerates the affected modules and integrates them in the deployed application.

In certain cases, the platform can also provide the capability for automated change management of the application; that is, it allows the facility of reconfiguring a deployed, running application *without looking at the application source code*. This requires twofold support, both at the time of application development and after the application is deployed. It involves the anticipation of possible future modifications to business process or application logic and the creation of appropriate customization points in the application at the time of application development. The eGovernance platform enables development of *easy-to-customize* applications by providing a *solution customizability enabler.* The accompanying wizard helps the application developer to identify the portions of business logic that may be reconfigured by the end user in order to specify the configurable parameters, and the processing required to translate the reconfigured logic into application code. An associated tool provides intuitive interfaces to help the end user modify the application after deployment. It allows the end user to modify the parameter values to suit the changed process, policy, or standard, and automatically performs the required modification to the application logic.

There is yet another way in which the framework helps a business user manage the eGovernance applications. Some of the building blocks of the framework separate the business rules and operational policies from the application logic. For example, the record management enabler encapsulates the data administration policies, the security enabler captures the access control policies, and the payment enabler creates payment policies. These policies can be changed without requiring changes to application source code. An asset may also use externally defined rules in a process. This provides the capability for *online* change management of the application, thus allowing the reconfiguration of a deployed, running application *without any downtime*.

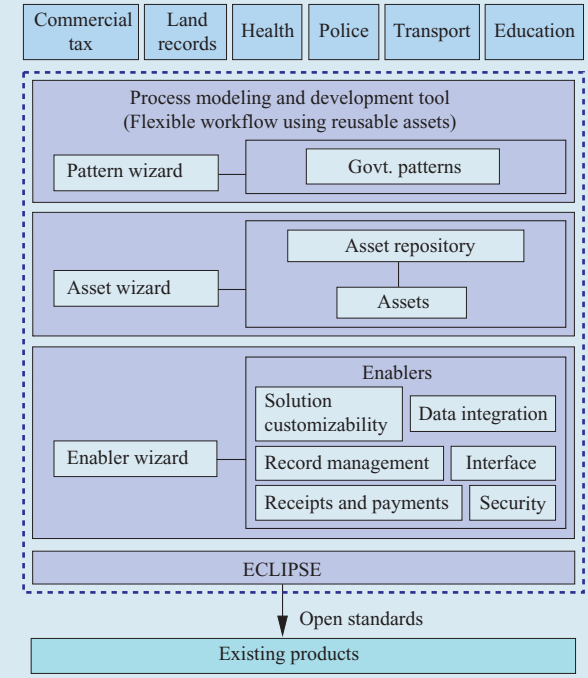

Figure 3

High-level architecture of the eGovernance framework.

5. Framework architecture

Figure 3 shows the high-level architecture of the eGovernance framework, based on the framework components described in Section 3. The top of the figure shows various government applications such as commercial tax, land records administration, health, police, transport, and education. The process modeling and development tool provides the capabilities for integrating citizens, employees, and applications across government agencies for the solution environment layer. It allows the solution developers or administrators to either create a new process flow or choose one from an existing set of eGovernance patterns which capture the typical high-level semantic process flow. For example, to create a renewal of driving license solution as described in Section 1, the solution creator chooses the lodgement with payment pattern. This pattern includes the following steps: citizen authentication and access control to the government

portal, application form with multiple language support, online data input including payment details, electronic form submission with digital signatures, electronic acknowledgment, input validation, automated workflows, electronic status tracking, e-mail notification of results, and generation and dispatch of necessary documents.

The process flow can be further customized, using the pattern wizard, by providing the details of each step in the process. The developer can either implement a process step from scratch using the existing development tools at a low level, or use the framework services described in Section 3, which allow application development at a higher semantic level. Each service caters to a requirement common across multiple government applications. The framework allows the packaging of these common services as assets, and facilitates the customization and reuse of these assets by multiple vendors. The solution developer can use the asset wizard to search for an asset on the basis of category names, select the asset, and view it in the appropriate viewer. For instance, the address verification Java** application requires a different viewer than the citizen data model. Examples of assets relevant to the driving license renewal solution are multilingual interfaces, online payment processing, digital signature verification, the address verification process, and the citizen data model. An asset can be a front end to an existing application or solution component providing the desired functionality, or it can provide the functionality by native implementation. For example, the online payment processing asset could be a wrapper to a legacy payment application, based on open standards such as Java Connector Architecture (J2C) [19]. Similarly, the digital signature verification asset can use third-party libraries for the public key infrastructure based on open security standards.

Underlying the high-level semantic modeling of eGovernance solutions using the assets is the support provided by the enablers—the development tools to build the reusable assets and customize these to suit the needs of a particular solution. The customization of an asset may be done manually using the edit tools provided in the framework, or it may be guided by an accompanying wizard. Guided customization works only for configurable assets. The framework provides tools to facilitate the creation of configurable assets and their accompanying wizards. For instance, in the address verification asset, the data source table name and the data fields to be verified may be configurable. Similarly, the digital signature verification asset could allow the developer to choose the digital signature algorithm, the key size, etc. An XML-based configuration language is defined to capture the configurable features of an asset. The wizard uses the configuration file of the asset to guide the developer through its customization. Figure 3 shows the various enablers described in Section 3. The solution customizability enabler was discussed in the section on solution management.

Implementation approach

The approach described in Figure 3 reuses several existing tools and products to build the framework. The framework was developed on the open source Eclipse [14, 15] platform. The Eclipse platform is built on a methodology for discovering, integrating, and running modules called *plug-ins*. Each tool provider develops its tool as a separate plug-in and exposes its tool-specific user interface (UI) in the Eclipse platform workbench. When the platform is launched, the user is presented with an integrated development environment (IDE) composed of the set of available plug-ins. A simple tool can be a single plug-in, while more complex tools may consist of many separate plug-ins. The Eclipse component architecture allows developers to easily integrate tools developed by multiple vendors.

The process modeling and development tool shown in Figure 3 provides a unifying wrapper for new and existing tools, assets, and solution components. The developers define a high-level business process model. Different modeling languages are used in the existing tools to capture various aspects of application modeling. For example, WebSphere Message Queue Workflow [11] uses its own workflow modeling language. WebSphere Studio [3, 4] supports BPEL [16] for process choreography, and the Struts [22, 23]-based Web application editor, WBI Modeler [11], has its own business processing modeling language. RAD builder [24] stores the Web application model in a different format, and Rational XDE [1] is based on unified modeling language [25]. Many of these tools are based on the Eclipse platform. The framework attempts to unify some of the existing modeling language plug-ins to provide an integrated high-level semantic modeling tool. Specification of the various steps in the pattern and mechanisms to reuse it are being devised and will evolve with the future customer engagements.

The process components in the framework service layer are implemented as a combination of Web-based application, enterprise Java development, connection to legacy applications, messaging, Web services, workflow definition, etc. Each component may use reusable assets and may be used as an asset itself. The framework could reuse the asset packaging and repository capabilities provided in the Rational XDE [1], based on the reusable asset specification [26]. The framework also provides common services and data models. It currently has data models for citizen, land, and establishment from past experiences. These are being further refined with the inputs from ongoing customer engagements. Other data models that have been identified from customer discussions include social welfare schemes, agricultural

commodities, property tax, etc. The common services over these data models and common government processes are evolving with real customer engagements. Some of the common processes identified thus far are described in the next section.

For the enabler layer, the framework reuses existing Eclipse-based tools and builds Eclipse plug-ins to interface with existing non-Eclipse tools. New plug-ins will also be developed to fill any gaps in the existing tools as new requirements emerge from early deployment. Currently, the interface enabler allows the development of conversational interfaces using the Eclipse-based WebSphere Voice Toolkit [27, 28]. The technology has been extended to support speech recognition for the Indian, English, and Hindi languages. The multidevice enabler supports the development of interfaces to different devices based on the Everyplace Toolkit [29, 30]. The record management enabler has been developed for the IBM DB2 database. It includes the Policy Driven Database Administration (PDDA) [31] engine to provide record management service. The technology provides a tool for policy specification and business object creation as well as runtime support to execute high-level policies. It enables government officials to specify the data administration and operational policies of their departments at a higher level in terms of business objects and their attributes. These officials are usually not IT experts and are thus not familiar with the task of database administration. For example, the Secretary of the Ministry of Railways can formulate the data administration and operational policies of the Ministry in terms of intuitive policy language and terms relevant to domain semantics, such as *refund policy dependent on window relative to train departure and approvals required.* A domain expert defines the mapping between the objects and the underlying databases of the eGovernance application once, and all application developers and solution administrators can subsequently take advantage of it. The policy maker need not have any knowledge about the underlying database schema or about database administration. Furthermore, while defining policies, it is very natural for the policy maker to specify that the policy should be executed only in a particular context. For example, an archival process may have to be executed only on the next holiday. The system supports such policies by associating various contexts with the policies. It has a predefined set of contexts such as *WEEKEND, AFTER OFFICE HOURS,* and *HOLIDAYS.* The policy maker is also allowed to define a new custom context; this enables the policy maker to incorporate the organizational calendar during policy specification. Finally, these policies are automatically deployed and executed at the appropriate times by the PDDA execution engine. The abstract architecture of autonomic database administration based on policies is shown in **Figure 4**. For further details, refer to [32, 33]. The solution customizability enabler is in the requirements analysis phase. The receipts and payments enabler will be developed by reusing the capabilities in WebSphere Commerce Payments [33]. The remaining enablers will evolve with actual customer engagements, described in the next section.

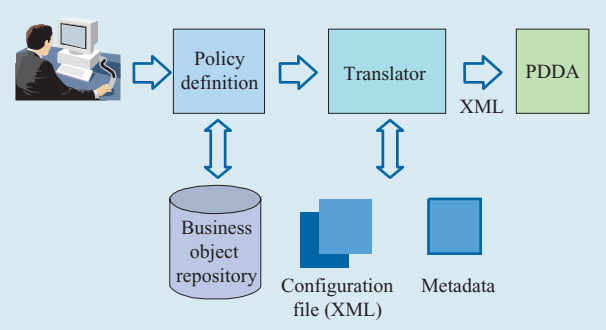

Figure 4

Policy-Driven Database Administration (PDDA) architecture.

Customer engagement experience

This section describes the insights obtained from past and ongoing customer engagements. We have been in discussions with several state governments and government organizations at the IBM India Research Laboratory. One of the proofs of concepts developed for a customer successfully demonstrated the capabilities of the record management enabler and the speech interface enabler. It was a farmer-oriented project for rapid collection and dissemination of market information to facilitate its efficient and timely utilization. The objective was to computerize market-related information (market fee, market charges, total arrivals, prices, storage, dispatches with destination, mode of transportation, costs, sold and unsold stocks, etc.). The information was collected at the local market sites and sent to a central site for consolidation. The consolidated information was then made available to the producers, traders, and consumers to enable them to derive maximum benefits from their transactions. Using the record management enabler, business objects were defined to represent the market commodities. The wizard was used to specify policies such as the following:

1. Notify the market site administrators whenever some new commodity is added at the central site.
2. Notify the market site administrator whenever the minimum sales price of a commodity is less than the specified minimum selling price for that commodity.

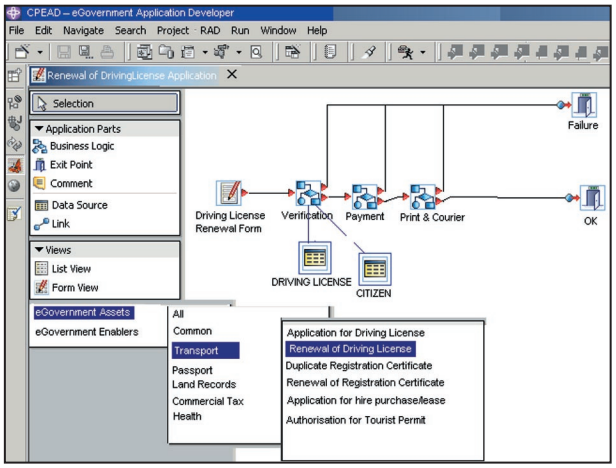

Figure 5

Categorization of eGovernment assets — driving license renewal.

3. Back up data for perishable commodities on a daily basis and for nonperishable commodities on a weekly basis.

The speech interface enabler was used, by illiterate farmers using a conversational interface, to navigate the consolidated market information on the Web. It used the technology for speech recognition in Hindi, the local language. There is a requirement, however, to do speech recognition for the various dialects of Hindi and a multitude of other local languages used in India. The customer also wants to support dynamic text-to-speech generation for various local languages. These are some of the market needs that must be addressed.

In an ongoing customer discussion, there is a requirement to measure the effectiveness of various social welfare schemes by a composite index and track the progress of each individual at each phase in his lifetime. Currently the services are delivered separately by each involved department, and individual metrics are used to measure the effectiveness of the scheme. The data is not captured effectively at the point of service delivery and is not consolidated across departments. We have submitted an approach document based on our eGovernance framework to address the customer requirements, and the initial response is very positive. The proposed approach includes a multidevice data capturing process based on the multidevice enabler, a consolidated data layer across different departments based on the record virtualization and data integration enabler, a lifecycle tracking process spanning various departments and different phases in a citizen's lifetime, and specification of the composite index, target group characteristics, access policies to consolidated data, and metrics to determine impact in an easily modified manner using the solution customizability enabler. The lifecycle tracking process can be designed as a reusable asset to address similar requirements by the health department, the urban development department, etc.

A pattern appears to emerge from the customer requirements described above, namely for data collection from various field offices and transmission (offline or online) to a central office for consolidation, processing, and analysis. The pattern is not limited to the scenarios described above and is applicable in tax returns, allotment of civil supplies, etc. The section on middleware framework services and reusable assets lists some of the other patterns identified while studying the government processes in past customer engagements.

In yet another ongoing customer engagement, we are in the process of studying the various department processes to identify the commonalities for reuse. Our initial understanding reveals the following candidates as reusable assets or framework services—the unified payment system, the complaint management system, the billing system, the utility approval process, various acknowledgment forms, rural and urban household data models, application forms for various licenses, etc. As mentioned earlier, the objective is to evolve the proposed eGovernance framework—tools, services, and solutions—to address the real customer requirements.

6. User interface and sample scenarios

This section describes the user interface of the eGovernance framework using a sample scenario. The starting point for rapid application development using the framework is the selection of the domain. The framework has been described for the eGovernance domain; however, it can be generalized to other domains such as banking, insurance, retail, and telecommunications. Each reusable component is associated with one or more domains. The selection of the domain by the developer creates an IDE that is customized for the selected domain. The remaining section describes the scenario for developing a driving license renewal application. The scenario shows how a developer can reuse configurable assets and enablers to rapidly create a new application using the framework.

To develop the application for renewal of a driving license, the developer can select the lodgement with payment pattern from the list of government patterns and search for the relevant assets to customize the pattern. The developer may find an application template that can be rapidly customized for this specific need or may use individual assets to customize the various steps of the

process flow. **Figure 5** shows that the assets can be categorized as belonging to various government departments such as Transport, Passport, Land Records, Health, and Commercial Tax. If required, the category names and asset organization can be customized by the developer. An asset may include processes, implemented applications, requirements documents, UML class diagrams, etc. For example, the Common category may include assets such as Registration, Log-in, Payment, Procurement Requirements and Test Plan, and Fund Monitoring Class Diagram. Similarly, as shown in Figure 5, the Transport department includes assets such as Renewal of Driving License, Renewal of Registration Certificate, and Authorization for Tourist Permit. The developer can click on any asset to see its details. The details of the Renewal of Driving License asset, shown in Figure 5, include a driving license renewal form that is processed by the verification logic using the citizen and driving license data sources, followed by the payment logic and printing and courier of the renewed driving license. The developer can customize the application template for a specific implementation. It is also possible for the developer to reuse the building blocks such as address verification, medical certificate verification, payment, the citizen data model, and the associated class diagrams to build a passport renewal application and save the asset under the Passport category. In the next set of screens, some components of the driving license renewal application are described in detail.

The developer can click on the driving license renewal form to see its contents in detail, as shown in **Figure 6**. There is a list of eGovernment custom tags that capture the common form elements used in the eGovernment applications: Address, Date, Class of Vehicle, Applicant Declaration, etc. The developer can drag and drop these custom tags on the application form. Each custom tag comprises a plurality of form fields that map to one or more data sources. For example, the Address tag comprises fields such as Street 1, Street 2, City, State, Country, and Pin Code from the Citizen data source. The developer can further customize the tag, for instance by deselecting the field Country from the Address tag for a specific implementation. Similarly, the Class of Vehicle tag refers to the various types of vehicles in the Driving License data source. **Figure 7** depicts the specification of user interfaces in local languages. It shows how the developer can specify the caption for the Address tag and its constituents in different languages. It also shows that the developer can customize the label of the field for each implementation. Further, the developer can use the interface enabler and the multidevice enabler for appropriate customization based on the deployment environment.

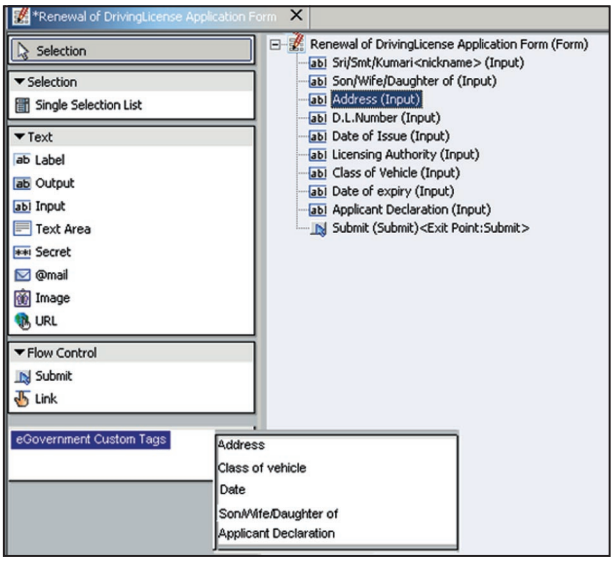

Figure 6

Driving license renewal form — eGovernment custom tags.

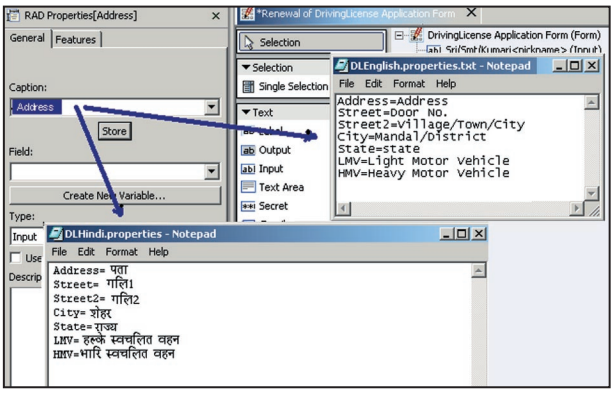

Figure 7

Specification of user interfaces in local languages.

The developer can click on the Verification logic block, shown in Figure 5, to see its contents. The business logic in an application can be modeled at a high level using building blocks such as relational database operations (RDB action), inter-application messaging, Web service invocation, connection to legacy systems, and business process, including staff events and code snippets. These building blocks are empowered by the appropriate enablers or tools for low-level code generation. For example, the RDB action may use the record virtualization enabler to achieve data integration across disparate databases. The Verification block, as shown

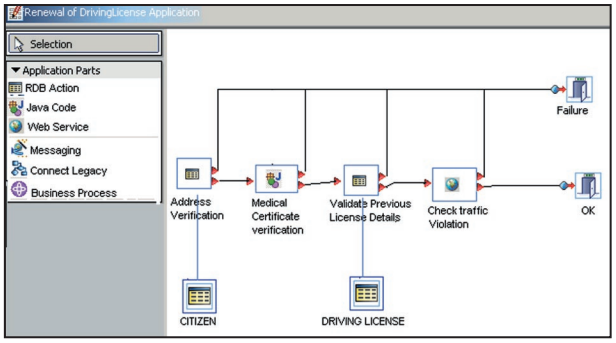

Figure 8

Verification logic block — high-level building blocks.

in **Figure 8**, comprises an RDB action to verify the applicant's address, Java code to verify the medical fitness certificate, another RDB action to validate previous license details, and a Web service invocation to check traffic violation details. These building blocks can be customized for a specific implementation of the driving license application or reused in another government application. The framework allows the specification of logic with no knowledge of standards required, since the high-level building blocks generate applications that conform to appropriate standards. For instance, the messaging building block generates code complying with the Java Messaging Standard (JMS) [20, 21], and the legacy connectivity is based on Java Connector Architecture (J2C) [19].

An asset in the framework can be saved either as a template or as an application. A template is a configurable component that can be converted to a customized application by a developer. Each building block in the template business logic is labeled as mandatory or optional. The optional building block can be deselected at the time of template customization. Further, a building block may also support configurable parameters whose values can be specified during customization. For example, in the Verification logic template, the Validate Previous License Details block, shown in Figure 8, may be optional and allow the parameters to be validated from the previous driving license to be configurable. The developer can choose to validate one or more parameters such as the driving license number, the applicant's date of birth, the date of expiration, or the class of vehicle. The template logic can generate customized application code based on the dynamic selection of these parameters. Similarly, the Check Traffic Violations block may be mandatory and may support different implementations such as RDB action or messaging-based, J2C-based, or Web-service-based implementations. The developer can select the implementation type and specify the appropriate configurable parameters. To develop such customizable components, both for ease of development and for ease of change management after deployment, the solution customizability enabler may be used.

Other enablers such as the receipts and payments enabler, the security and privacy enabler, and the record Management enabler can also be used to create the required building blocks. These building blocks and the associated services constitute an integral part of the application being developed.

7. Related work

The development, deployment, and management of complex, integrated solutions respectively require highly skilled developers, experienced administrators, and IT-knowledgeable business users, primarily because of the complexity involved in working at a low level of abstraction and dealing with multiple proprietary technologies.

Vassilakis et al. [34] propose an approach to handling electronic service lifecycles that balances responsibilities between domain experts and IT professionals. This approach enables a more holistic management of the electronic service lifecycle by employing modeling and representation in high levels of abstraction and incorporating tools for automatically generating operative service instances from these high-level descriptions. Several application development tools such as WebSphere Studio Application Developer [3, 4] allow rapid development of J2EE-compliant applications [16, 17] based on open standards such as XML [35] for data interchange, J2C [19] for application connectivity, and JMS [20, 21] for messaging. Tools such as StrutsWizard [2] simplify the development of applications based on design patterns such as Struts [22, 23], and other tools such as DB3NF [36] allow easy creation of scalable Web applications based on Microsoft technologies. The WebSphere Voice Toolkit [27, 28] helps create VoiceXML* applications rapidly, using an integrated development environment. The Everyplace Toolkit [29, 30] allows developers to quickly and easily develop Web applications targeted to multiple devices with different characteristics. Though the proposed approaches and tools do simplify application development to some extent, they do not address all of the issues involved in developing eGovernance applications as described in this paper. They support standards for low-level requirements such as connectivity, messaging, and data exchange, but the need to abstract high-level building blocks such as security, data integration, and record management is not addressed. As described earlier, the task is even more challenging for

eGovernance solutions because of the scale, higher stakes in security, lack of reliable infrastructure, budgetary constraints, and low level of IT awareness among government officials and the common public.

At present, software vendors offer eGovernance solutions by integrating existing products and by customizing those for a specific eGovernance application [5–8]. The existing solutions also require customization to adapt to regional languages, local tax structures, and government policies. This customization is done using proprietary technologies, data schemas, and standards. This not only creates islands of applications that do not inter-operate, it also limits the reuse of solution components across different government applications, leading to duplication of efforts and increased development costs. The concept of reusable assets or solution components has been detailed very well in the Reusable Asset Specification [26]. Tools such as Rational XDE [1] conform to this specification. However, the scope of an "asset" is limited to packaging the related artifacts together and providing support to search these assets and use them as is. Solution components such as the processes very often require customization. Application templates, by their very definition, require the developer to specify some parameters to develop an application. The WebSphere Studio [4, 24] does introduce a set of templates, each of which stores a description of an application along with customized settings. These settings enable wizard-driven specification of parameters and automatic generation of code to produce a complete operational application. Further, an associated tool, the RAD builder [24], enables the development of new application templates. However, the template language is proprietary, and the customizability is limited to HTML tags and relational database (RDB) operations. The tool does not support the development of customizable processes or business logic other than RDB operations. Moreover, there is no emphasis on the development of easily reconfigurable solutions to enable change management once the solution is deployed.

A tangential but related effort is described in [37] to understand new models of collaboration for delivering government services. While that study is more focused on the kinds of services and business relations required, the proposed framework can possibly provide the technical infrastructure for the delivery of the same. The network-infrastructure-related efforts for digital government are described in [38], which proposes a scalable architecture for collection of data over wide-area networks. The E-Rulemaking concept described in [39] can be another potential service to be offered by the eGovernance framework, in addition to the services described in this paper.

The framework described in this paper attempts to address the various gaps in the offerings available today, specifically in the light of government applications and multivendor development.

8. Conclusions and future directions

This paper has described the eGovernance framework, an eGovernance solution development platform that will lower the cost of developing, deploying, and managing government solutions. The framework provides repositories of solution components such as security handlers, record management components, and user interface components. It also provides data models for entities such as citizens, businesses, and establishments, and repositories for actual data corresponding to these entities so that all eGovernance applications can share them. The solution components are customizable for each solution independently through a wizardlike interface.

The solution components and data models in the framework are described at a higher semantic level, and they are built with customization points that can be programmed through a policy administration interface which is fairly intuitive and intended for solution managers who may not be well versed in application development. It also allows applications to be easily modified after deployment to reconfigure the existing eGovernance process in response to government policy changes. The platform also enables sharing of development effort across applications developed by multiple vendors through reuse of assets.

Acknowledgments

The authors wish to thank Dilip Antony Joseph, Manish Bhide, Ajay Gupta, Mukul Joshi, and Shree Raman for their contribution in developing the policy management aspects of the framework.

References

1. Rational XDE; see *http:www.ibm.com/developerworks/rational/products/xde/*.
2. StrutsWizard; see *http://www.strutswizard.com/index.shtml*.
3. U. Wahli, I. Brown, F. Ferraz, M. Schumacher, and H. Sjostrand, *WebSphere Studio Application Developer Version 5 Programming Guide,* ISBN: 0738499579, 2003; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG246957/*.
4. WebSphere Studio Application Developer (WSAD); see *http://www-3.ibm.com/software/awdtools/studioappdev/*.
5. Arkansas Office of Motor Vehicles; see *http://www-3.ibm.com/software/success/cssdb.nsf/CS/JMAY-5DHNGF?OpenDocument&Site=software/*.

6. California Franchise Tax Board; see *http://www-3.ibm.com/software/success/cssdb.nsf/CS/JMAY-5DHPAP?OpenDocument&Site=software/*.
7. Ministry of Justice uses Web-Services; see *http://www-3.ibm.com/software/success/cssdb.nsf/CS/LEOD-5KJV36?OpenDocument&Site=software/*.
8. Rational Rapid Developer Delivers J2EE applications at Large State Regulatory Agency; see *http://www-3.ibm.com/software/success/cssdb.nsf/CS/JENS-5SBJF2?OpenDocument&Site=software/*.
9. e-Government Interoperability Framework; see *http://www.govtalk.gov.uk/schemasstandards/egif_document.asp?docnum=731/*.
10. *IBM Endowment for the Business of Government, E-Government 2003,* M. A. Abramson and T. L. Morin, Eds., Rowman & Littlefield Publishers Inc., Lanham, MD, 2003.
11. L. Gavin, G. Diederichs, P. Golec, H. Greyvenstein, K. Palmer, S. Rajagopalan, and A. Viswanathan, *An EAI Solution using WebSphere Business Integration (V4.1),* ISBN: 0738426547, 2003; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG246849/*.
12. M. Galic, A. Halliday, A. Hatzikyriacos, M. Munaro, S. Parepalli, and D. Yang, *A Secure Portal Using WebSphere Portal V5 and Tivoli Access Manager,* ISBN: 073849853X, 2003; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG246077/*.
13. W. D. Zhu, S. Jefferson, M. Adair, M. Pepper, and H. Martens, *Content Manager On Demand Guide,* ISBN: 0738429422, 2003; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG246915/*.
14. Eclipse; see *http://www.eclipse.org/*.
15. E. Gamma and K. Beck, *Contributing to Eclipse: Principles, Patterns, and Plugins*, First Edition, ISBN: 0321205758, Addison-Wesley Publishing Co., Inc., Reading, MA, 2003.
16. Java 2 Platform, Enterprise Edition (J2EE); see *http://java.sun.com/j2ee/*.
17. R. Johnson, *Expert One-on-One J2EE Design and Development*, ISBN: 0764543857, Wrox Press, John Wiley & Sons, Inc., Hobokon, NJ, 2003.
18. S. Chatterjee and J. Webber, *Developing Enterprise Web Services: An Architect's Guide*, ISBN: 0131401602, Prentice-Hall, Inc., Englewood Cliffs, NJ, 2003.
19. J2EE Connector Architecture (J2C); see *http://java.sun.com/j2ee/connector/*.
20. Java Message Service API (JMS); see *http://java.sun.com/products/jms/*.
21. S. Terry, *Enterprise JMS Programming*, First Edition; ISBN: 0764548972, John Wiley & Sons, Inc., New York, 2002.
22. T. Husted, C. Dumoulin, G. Franciscus, D. Winterfeldt, and C. McClanahan, *Struts in Action: Building Web Applications with the Leading Java Framework*, ISBN: 1930110502, Manning Publications Company, Greenwich, CT, 2002.
23. Struts Framework; see *http://jakarta.apache.org/struts/*.
24. Rapid Application Technologies in WebSphere Studio; see *http://www7b.software.ibm.com/wsdd/zones/studio/rad/*.
25. J. Arlow and I. Neustadt, *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, ISBN: 0201770601, Addison-Wesley Publishing Co., Reading, MA, 2001.
26. Reusable Asset Specification; see *www.rational.com/ras/index.jsp/*.
27. R. Credle, G. Kempny, S. Dadhich, E. Dietrich, J. Hu, and J. Poggioli, *IBM WebSphere Voice Systems Solutions,* ISBN: 073842773X, 2003; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG246884/*.
28. Websphere Voice Toolkit; see *http://www.alphaworks.ibm.com/tech/voicetoolkit/*.
29. Everyplace Toolkit for WebSphere Studio; see *http://www-3.ibm.com/software/pervasive/products/mobile_apps/everyplace_toolkit.shtml*.
30. J. Rodriguez, E. Forestier, R. Guru, G. Kroner, L. Patterson, H. Tran, A. Venancio, and G. Villavicencio, *WebSphere Everyplace Access Version 4.3 Handbook for Developers,* ISBN: 0738498750; IBM Redbook, see *http://www.redbooks.ibm.com/redbooks/SG247015/*.
31. D. A. Joseph, M. Mohania, M. Kumar, M. Bhide, A. Gupta, and M. Joshi, "Defining Data Administration and Operational Policies at the Business Object Level for E-Governance Applications, *Proceedings of the First International Conference on E-governance,* December 2003; see *http://dilu.tripod.com/research/projects/*.
32. M. Bhide, S. Pandey, A. Gupta, and M. Mohania, "Dynamic Access Control Framework Based on Events," *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, India, 2003, pp. 765–767.
33. WebSphere Commerce Payments; see *http://www-306.ibm.com/software/genservers/commerce/payments/lib.html*.
34. C. Vassilakis, G. Laskaridis, G. Lepouras, S. Rouvas, and P. Georgiadis, "A Framework for Managing the Lifecycle of Transactional e-Government Services," *Telematics & Informatics* **20,** No. 4, 315–329 (2003).
35. M. J. Young, *XML Step by Step*, Second Edition, ISBN: 0735614652, Microsoft Press, Redmond, WA, 2001.
36. DB3NF; see *http://www.db3nf.com/overview/*.
37. S. S. Dawes and L. Prefontaine, "Understanding New Models of Collaboration for Delivering Government Services," *Commun. ACM* **46,** No. 1, 40–42 (2003).
38. L. Golubchik, W. C. Cheng, and C. Chou, "Bistro: A Scalable and Secure Data Transfer Service for Digital Government Applications," *Commun. ACM* **46,** No. 1, 50–51 (2003).
39. J. E. Fountain, "Prospects for Improving the Regulatory Process Using E-Rulemaking," *Commun. ACM* **46,** No. 1, 63–64 (2003).

Received December 5, 2003; accepted for publication March 1, 2004; Internet publication September 16, 2004

Parul A. Mittal *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (mparul@in.ibm.com).* Ms. Mittal is a Research Staff Member at the IBM India Research Laboratory, Delhi. She holds a B.Tech. degree in electrical engineering from the Indian Institute of Technology (IIT), Delhi and a Master of Science degree from the University of Michigan, Ann Arbor, in computer science. Prior to joining IBM, she worked at Hughes Software Systems in India. Her primary research interests are in the area of eCommerce, knowledge management, and eGovernance.

Manoj Kumar *IBM Corporate Division, Route 100, Somers, New York 10589 (manoj1@us.ibm.com).* Dr. Kumar is the Director of Corporate Technology Evaluation at IBM. He served as the Director of the IBM India Research Laboratory from 2000 to 2003. From 1983 to 2000, Dr. Kumar worked at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, where he led projects in electronic commerce, video delivery and encoding technologies, and the architecture and design of processors and parallel computers. Dr. Kumar received his B.Tech. degree from the Indian Institute of Technology in Kanpur, India, in 1979, and his M.S. and Ph.D. degrees from Rice University in 1981 and 1984 respectively, all in electrical engineering.

Mukesh K. Mohania *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (mkmukesh@in.ibm.com).* Dr. Mohania received his Ph.D. degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 1995. He is currently a manager at the IBM India Research Laboratory. His areas of interest are distributed databases, data warehousing, semi/unstructured databases, XML data integration, data mining, and autonomic computing. He has published more than 75 research papers in these areas in leading international journals and conference proceedings, and as book chapters. He has organized several international conferences and workshops as program chair and has edited several conference proceedings. In 2000, Dr. Mohania received a Technical Achievement Award in the area of Web database management and data warehousing from the Association of Database and Expert Systems Applications in Greenwich, U.K. He is a Senior Member of the IEEE.

Medha Nair *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (nmedha@in.ibm.com).* Ms. Nair is a software IT architect with the Software Group of IBM India. She holds B.E. and M.E. degrees in computer science from the Shri Govindram Seksaria Institute of Technology and Science, Indore. She has 15 years of industry experience, more than seven of them as a software architect in pre-sales. She works primarily in the area of eGovernance in India.

Nipun Batra *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (nipbatra@in.ibm.com).* Mr. Batra is a Technical Staff Member at the IBM India Research Laboratory, Delhi. He holds a B.Tech. degree in production engineering from Pune University. Prior to joining the IBM Research Laboratory, he worked with IBM Global Services, Pune. He was also part of an aSAPXcess team with ASAP Solutions. Mr. Batra works primarily in the area of eCommerce and Eclipse technology.

Prasan Roy *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (prasan@in.ibm.com).* Dr. Roy is a Research Staff Member at the IBM India Research Laboratory, Delhi. He holds a B.Tech. degree from the Indian Institute of Technology, Delhi, and a Ph.D. degree from the Indian Institute of Technology, Bombay, both in computer science. Prior to joining IBM, he held faculty and research positions at IIT-Bombay and Bell Laboratories. He works primarily in the area of database systems, encompassing data integration, XML storage, query optimization, and transaction processing.

Anupam Saronwala *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (asaronwala@in.ibm.com).* Mr. Saronwala is responsible for managing research projects and business development activities at the IBM India Research Laboratory, New Delhi. Prior to joining IBM, he had more than 18 years' experience in software research and development, consulting and operations with leading U.S. and Indian companies. He holds a M.S. degree in computer engineering from Syracuse University, New York, and a B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur. His primary areas of interest are in introducing leading-edge technologies to emerging markets, specifically in the areas of pervasive computing and eGovernance.

Lalit Yagnik *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (yagnikl@sg.ibm.com).* Mr. Yagnik heads the e-business Software Center of IBM India, responsible for software technical solutions consulting. He holds a Master of Science (Technology) degree from the Birla Institute of Technology and Science in computer science and has more than 25 years' experience in consulting, architecting, and implementing software solutions. Joining IBM Australia in 1985, he has led clients with technical solutions and established software competency teams. Prior to his India assignment in 2000, he was Director of Software Services at SingaLab, a software R&D joint venture of IBM with the Singapore government. His interests include technology architectures for business innovation, eGovernance, database systems, integration, Web services, and technology–people–process synergy.