Develop decision services, Part 1: A smarter city case study

Skill Level: Intermediate

Raj Rao (rrao2@us.ibm.com)
BRMS IT Specialist
IBM

Sandeep Desai (sandeep@us.ibm.com)
Enterprise IT Architect
IBM

28 Jun 2011

In today's ever-changing business environment, there is a need for systems to be smarter to reduce the overall cost and complexity and to help businesses react quickly to changing environments. To do this, a smarter system needs an effective decision subsystem. In this three-part article series, we use a smarter city use case to guide architects and developers in selecting and implementing a decision subsystem. We walk you through a complete application development lifecycle of a business rule management system (BRMS). We discuss the BRMS architecture, rules discovery process, developing the rule application, and rule maintenance. The smarter city use case involves implementing a decision subsystem for intelligent coordination among multiple city departments during severe weather emergencies. The decision subsystem is based on IBM WebSphere® ILOG® JRules to automate the decision-making process.

Introduction

This article series is intended for architects and developers unfamiliar with a business rule management system (BRMS) and who want to gain an understanding of the process of creating a decision service. The article describes the rationale for choosing a BRMS and IBM WebSphere® ILOG® JRules. In addition, the BRMS architecture and integration with the enterprise architecture are outlined. For the developer, this article proposes an application development process for building the

rule application. Best practices and pitfalls to avoid are included throughout the article.

To make a system smarter, it needs to make timely decisions based on incoming events. A decision service is essentially a service that automates complex decision making by utilizing business rules. These business rules are declarative, are separate from the application code, and can be authored and managed by business users. Each of these business rules might be simple, but collectively they can make complex business decisions.

WebSphere ILOG JRules is the IBM technology for creating, maintaining, and deploying decision services and business rules. It enables business users to dynamically make changes and manage the rules based on business needs without involving IT.

WebSphere ILOG JRules offers a number of options at every stage of the application development process, which makes the tool very powerful and flexible but also poses a challenge for a newcomer to grasp the entire development process. In this article, we walk you through one typical path among the many possible ones.

The product versions used in this article are:

- WebSphere ILOG Rule Studio V7.1.1
- WebSphere ILOG Rule Execution Server V7.1.1

Familiarity with XML, Java™, and the Eclipse IDE is assumed. Review the Resources section for relevant links for additional information.

Smarter city overview

Scenario description

Our scenario city, Rotterdam, has built water plazas to prevent flooding resulting from heavy rainfall (see Resources). During dry periods, the water plazas serve as recreational areas, such as children's playgrounds or sports fields. During heavy rainfall, the water plazas are used to retain the excess rainwater. The city water department then discharges water into the local waterways in a controlled manner.

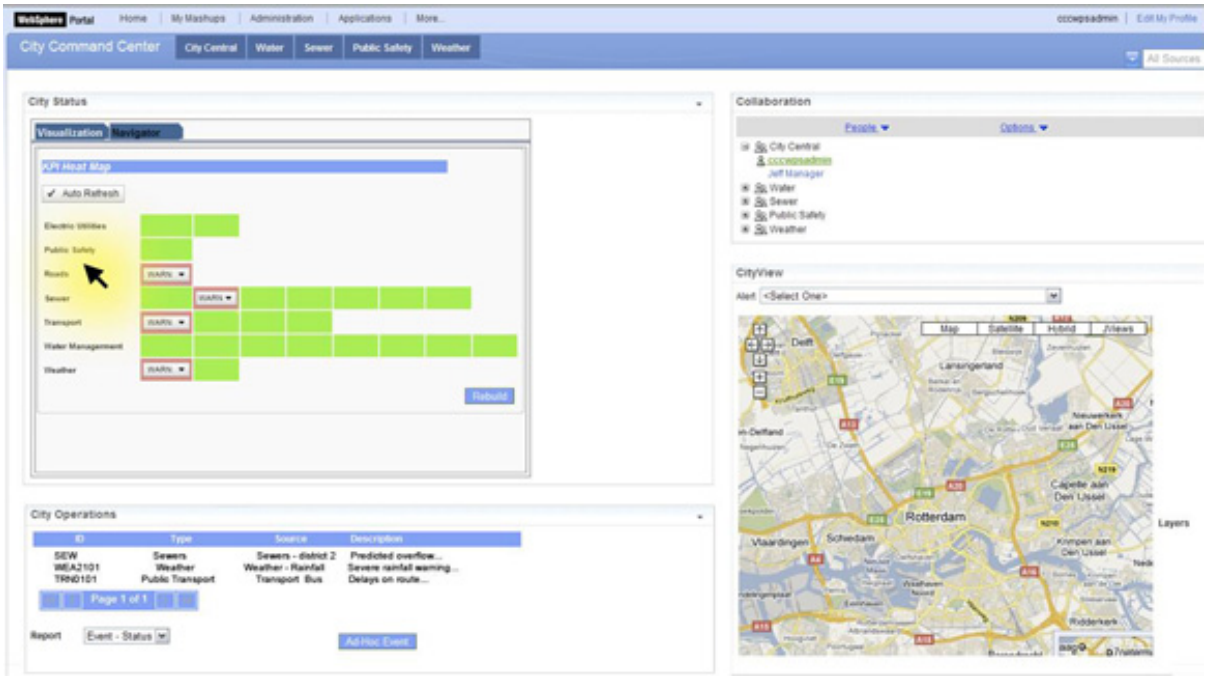
A smarter city command center monitors weather-related events, such as approaching heavy rainfall. The city staff must make decisions quickly and efficiently to proactively prevent the flooding of the city. They have to take into consideration a variety of factors:

- The amount of rainfall predicted

- The amount of rainfall already observed
- Current state of water plazas
- Public safety officials available to help evacuate water plazas
- Traffic situation around water plazas

They use a city portal as shown in Figure 1. The portal provides a command center dashboard view that displays information from various departments' IT systems. (View a larger version of Figure 1.)

Figure 1. Smarter city command center



Business requirements for the solution

In the smarter city command center scenario, many of the decisions made by city staff can be effectively automated using a decision service. Our scenario city has the following requirements for decision services:

1. Must be able to automate complex business logic
2. Offers role-based access for all participants in the rule management process
3. Must enable non-technical business users to maintain business logic
4. Must be able to easily make frequent changes to business logic without

the involvement of IT

5. Includes the ability to audit and trace business decisions
6. Provides accurate execution reports

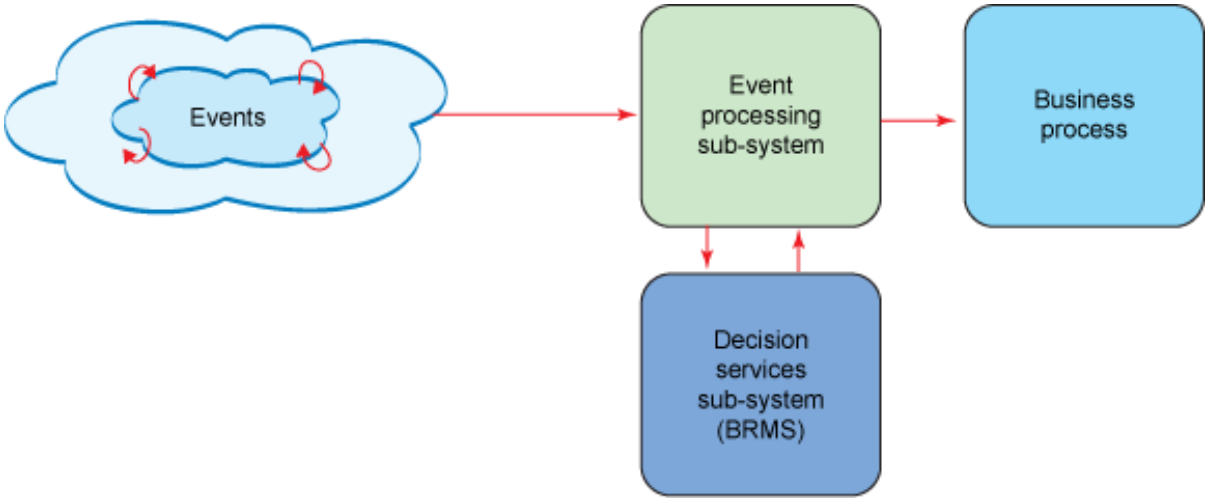
High-level solution

In our smarter city command center enterprise architecture, a BRMS is used to implement the decision service. It provides a central repository to define, modify, and maintain all business rules. Implementing a business rules subsystem using WebSphere ILOG JRules provides the business users, that is, the city command center staff, the ability to define and modify the business rules dynamically and on demand, without the intervention of IT staff.

WebSphere ILOG JRules allows us to implement the BRMS as a service that any other subsystem can invoke. It can be invoked using web services or service component architecture (SCA), thus enabling service-oriented architecture (SOA). In our scenario architecture, the decision service is invoked by an event processing subsystem using web services. If our enterprise uses an enterprise service bus (ESB), the same decision service can be invoked from the ESB using web services or SCA.

Figure 2 shows the smarter city command center solution architecture.

Figure 2. Solution architecture



Overview of WebSphere ILOG JRules

WebSphere ILOG JRules provides role-based modules, as shown in Figure 3.

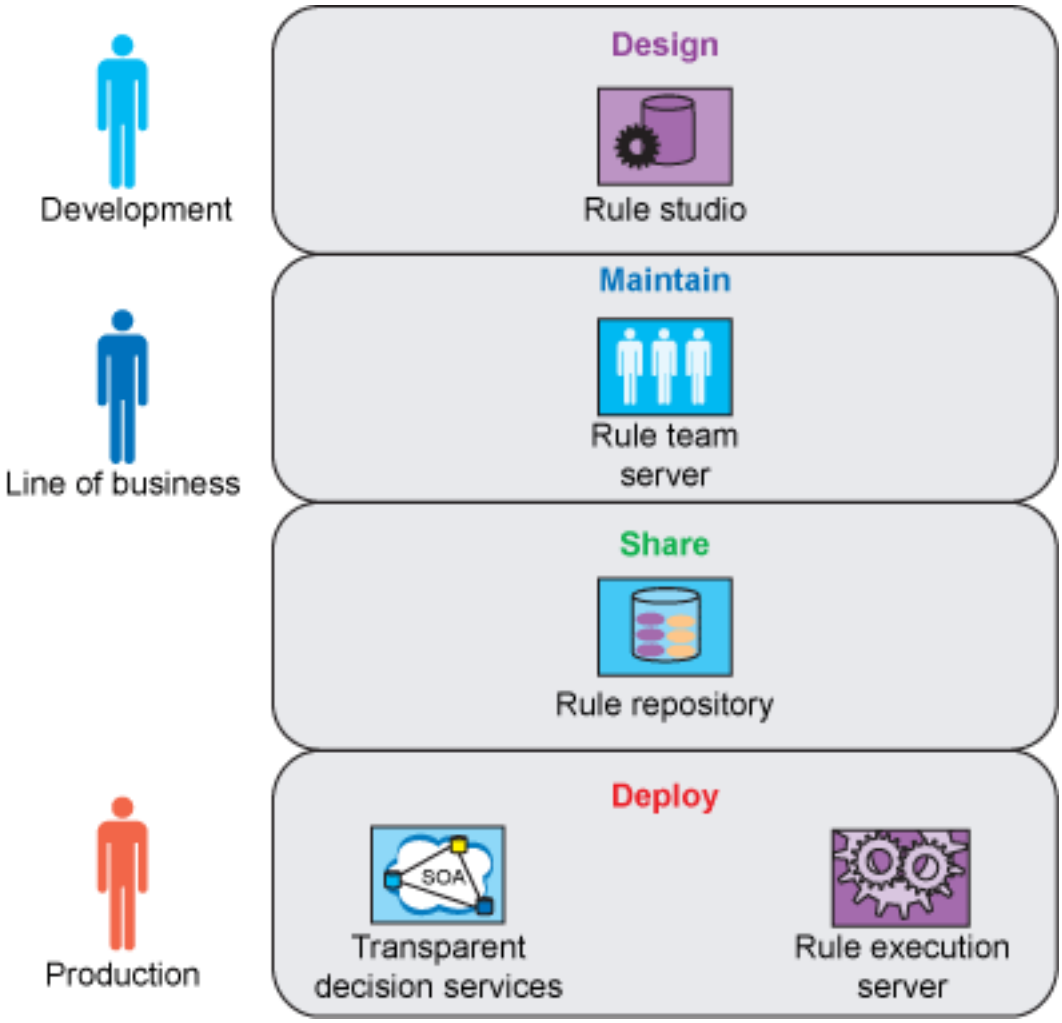
1. **IT developer**-- responsible for development of business rule applications.

Developers work with an Eclipse-based IDE called Rule Studio for design, Java development, and rule project development.

2. **Line-of-business user**-- responsible for business rule authoring and management.
   Business users work with a browser-based Rule Team Server to write and maintain business rules. Business users can also perform user testing and simulation on the Rule Team Server.

3. **Production administration**-- responsible for integrating, monitoring, and auditing the decision subsystem in an enterprise.
   Administrators have access to the Rule Execution Server to monitor deployed rulesets and manage decision services. In addition, they can use the Decision Warehouse to perform fine-grained auditing.

Figure 3. BRMS application development



Anatomy of the decision

The purpose of the decision service is to intelligently generate notifications and directives to various departments based on incoming severe weather events. The input and output to this rule engine conform to the Common Alerting Protocol (CAP), an XML specification produced by OASIS/ITU-T 9 (see Resources). The input data contains information about weather alerts, such as rainfall, and fog. Figure 4 shows the input fragment of a sample CAP XML.

Figure 4. Input XML fragment

```
<info>
   <language>en-US</language>
   <category>Met</category>
   <event>HeavyRainfall</event>
   <responseType>Assess</responseType>
   <urgency>Immediate</urgency>
   <severity>Severe</severity>
   <certainty>Observed</certainty>
   <expires>2010-11-15T16:00:00+00:00</expires>
   <senderName>NATIONAL WEATHER SERVICE ROTTERDAM NL</senderName>
   <headline>SEVERE THUNDERSTORM WARNING</headline>
   <description>AT 254 PM PDT...NATIONAL WEATHER SERVICE DOPPLER RADAR INDICATED A SEVERE THUNDERSTORM OVER ROTTERDAM CITY..
   <instruction>TAKE COVER IN A SUBSTANTIAL SHELTER UNTIL THE STORM PASSES.</instruction>
   <contact>CITY/WEATHERFCT</contact>
   <parameter>
      <valueName>RainfallLevel1H</valueName>
      <value>10</value>
   </parameter>
   <parameter>
      <valueName>RainfallLevel6H</valueName>
      <value>60</value>
   </parameter>
```

(View a larger version of Figure 4.)

Based on these alerts, the decision service makes decisions to generate notifications or directives for various city departments, as seen in the sample output fragment shown in Figure 5.

These intelligent notifications and directives are generated by business rules.

Figure 5. Output XML fragment

```
<sender xmlns="urn:oasis:names:tc:emergency:cap:1.1">CityCommandCenter-BRE</sender>
<sent xmlns="urn:oasis:names:tc:emergency:cap:1.1">2010-10-05T21:42:05.921</sent>
<info xmlns="urn:oasis:names:tc:emergency:cap:1.1">
   <language>en_US</language>
   <category>Safety</category>
   <event>AlertLevelRainfallAssessment</event>
   <audience>CityWaterDomain CityPublicSafetyDomain</audience>
      <valueName>Type</valueName>
      <value>Notification</value>
   </eventCode>
   <headline>Level Rainfall Assessment</headline>
   <description>Heavy rainfall expected; Assessing rainfall vs. sewer capacity</description>
   <contact>ccc@rotterdam.com</contact>
```

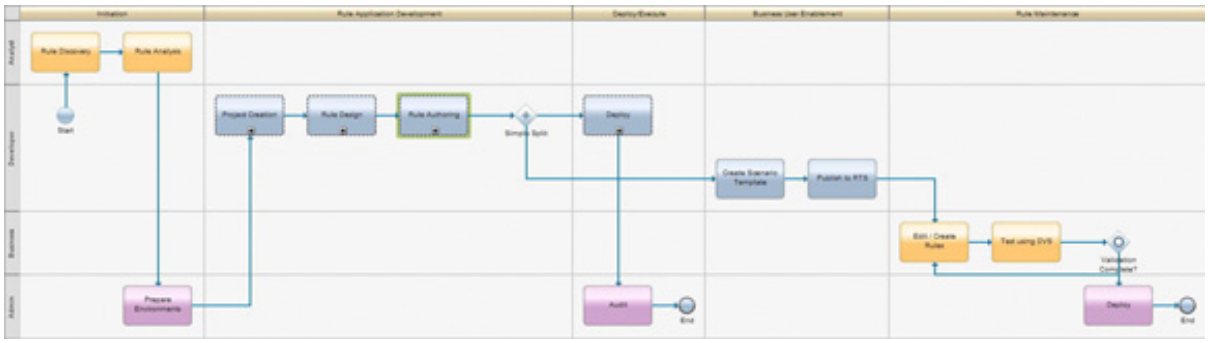
(View a larger version of Figure 5.)

Decision service creation process

A significant advantage of rule-based development is that it externalizes the business rules from the application code and decouples the development cycles. Rule-based development separates the application development and deployment cycle from the development and deployment track for rules. Therefore, the process for creating rule-based applications is different from traditional development processes. The process of creating a decision service is depicted in the process

diagram shown in Figure 6. As you can see in this process diagram, creating a decision service requires that rule analysts, rule architects, and business subject matter experts work together. (View a larger version of Figure 6.)

Figure 6. Process for creating the decision service



The tasks involved in the application development process are as follows:

1. Initiation
    a. Rule discovery
    b. Rule analysis
    c. Prepare environments
2. Development
    a. Project creation
    b. Rule design
    c. Rule authoring
3. Deploy and execute
    a. Deploy
    b. Audit
4. Enable business users
    a. Create scenario Excel template
    b. Publish to Rule Team Server
5. Rule maintenance

a. Edit/write rules

b. Test

c. Deploy

At a high level, the process starts with these steps:

1. The initiation phase, where the rule analyst discovers and harvests business policies.
2. Policies are then analyzed to create unambiguous business rules.
3. The rule developer uses Rule Studio to create rule projects and write the initial set of rules.
4. The rules are then deployed to the Rule Execution Server (RES), which leverages the Hosted Transparent Decision Services (HTDS) capability of WebSphere ILOG JRules to expose the ruleset as a web service.
5. To enable business users to maintain the business rules, rules are published to the Rule Team Server (RTS). Not only can business users write rules in RTS, but also they can test these rules from RTS using the Decision Validation Service (DVS).
6. When ready, rules are extracted and deployed to the RES.

Details of each of these processes are provided. But first, a few caveats are in order.

The process diagram shown in Figure 6 shows a typical process for creating a decision service, and it is indeed the process used in our case study. It is not meant to imply, though, that this is the only valid process or even the recommended process for all situations. For instance, auditing using a decision warehouse is an optional step that an organization might choose to skip.

This article is not intended as a detailed tutorial, but more as an example of the development process. It is intended to supplement, and not substitute for, the product documentation. When WebSphere ILOG JRules wizards are used in a task, however, details of the information entered into the wizard are provided. Also, WebSphere ILOG JRules offers a large set of capabilities and features, but this process does not attempt to touch on all these capabilities. For instance, we do not use rule solutions for Microsoft Office or EJB deployment in this case study.

Additionally, while Figure 6 depicts a sequential series of steps, rule development often benefits from an agile, iterative development cycle, which, in fact, is the

recommended approach. Within an iteration, however, this process can be applied after skipping some completed activities, such as preparing the environments or creating the projects.

This case study uses the default WebSphere ILOG JRules installation on the WebSphere Application Server Community Edition, and the steps related to installation and configuration of WebSphere ILOG JRules, including authentication and permissions management, are therefore not covered.

Discovering business rules

Business rules might be thought of as conditional statements that are used to make decisions. Business rules abound in any organization, and they are often locked up in the heads of subject matter experts. These rules are communicated in the form of business policies, such as these:

- *If rainfall is heavy, send directive to Sewer Department to assess rainfall versus sewer capacity.*
- *If water plaza is cleared and threat level is high, send directive to activate water plaza and send directive to Sewer Department to monitor water quality in the plaza.*

There are several dozens or hundreds of such policies in a typical ruleset. Each of these policies individually makes sense to a businessperson and can be comprehended in isolation, but they work together to make the business decision.

In the context of a BRMS, business rules are executable units; therefore, each of these policies needs to be refined to precisely capture the business rule, leaving no ambiguity. This refinement typically is a two-step process:

1. Perform rule analysis to determine rule groups and overall rule flow logic.
2. Write the business policies using terms defined in the business object model (BOM).

Analyzing the rules

Rule analysis involves carefully studying the policies to identify rule groups based on common patterns, organizing these policies in a structured manner, and formulating an overall flow of logic through these rule groups.

Let us consider one of the business policies:

If rainfall is heavy, send a directive to the Sewer Department to assess rainfall versus sewer capacity.

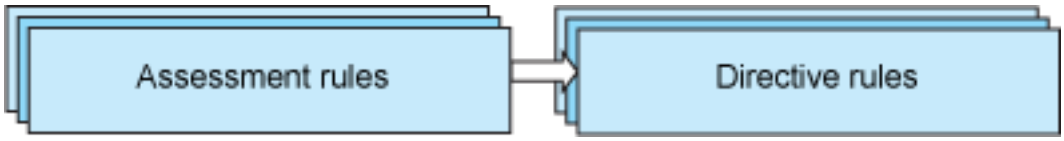
There is a lot of ambiguity in this policy. For instance, what constitutes "heavy rainfall"? It turns out that an assessment of "heavy rainfall" can happen based on several conditions, such as:

- Observed rainfall in the last hour is over 15 mm.
- Observed rainfall in the last hour is over 10 mm, and the observed rainfall in the last 12 hours is over 100 mm.
- Observed rainfall in the last 12 hours is over 150 mm.
- Observed rainfall in the last 6 hours is over 80 mm.

Each of these conditions can be independently expressed as a business rule. It is not uncommon to find that removing ambiguities from a business policy might actually result in several simpler, well-defined business rules.

We also notice a rule group here: rules that perform assessments based on observed alerts. Another group of rules generates notifications and directives based on inferred assessments. Figure 7 shows a logical ruleflow.

Figure 7. Logical rule flow



In the subsequent articles in this series, we describe the process for actually implementing these rules and ruleflows in WebSphere ILOG JRules.

Conclusion

We have seen that WebSphere ILOG JRules is a powerful tool that offers a rich set of features for building business rules management systems that can be executed in an SOA environment. Using a case study, we have walked through the rationale for choosing a BRMS and the high-level solution architecture. In addition, we have walked through the initiation phase of the decision service creation process where business policies are harvested and analyzed. In subsequent articles, we drill down into the development process for the technical developer and for the business user.

Resources

Learn

- Rotterdam water plazas: Our scenario city built water plazas to prevent flooding resulting from heavy rainfall.
- IBM WebSphere ILOG JRules product page: Access more about the features and benefits, system requirements, and support from the product home page.
- IBM WebSphere ILOG JRules Information Center: Find more information about this product line and its features.
- OASIS Common Alerting Protocol 1.1: The Common Alerting Protocol (CAP) is a simple but general format for exchanging all-hazard emergency alerts and public warnings over all kinds of networks.
- Policies and Rules – Improving business agility: Part 1: Support for business agility (Hondo, Boyer, Ritchie, developerWorks, March 2010): One challenge in architecting and implementing agility in business solutions today is that the use of the terms, policy, and rule, differs across products. Learn the concepts and relationships of policies and rules technologies to implement specific business strategies and tactics.
- Creating intermediate facts in WebSphere ILOG JRules using synthetic objects (Raj Rao, developerWorks, November 2010): This article uses a scenario to introduce the intermediate fact pattern in business rule processing and describes a technique in IBM WebSphere ILOG JRules to create intermediate facts using synthetic objects in the business object model.
- IBM developerWorks Industries: Get all the latest industry-specific technical resources for developers.
- developerWorks technical events and webcasts: Stay current with technology in these sessions.
- developerWorks on Twitter: Join today to follow developerWorks tweets.
- developerWorks podcasts: Listen to interesting interviews and discussions for software developers.

Get products and technologies

- WebSphere ILOG JRules V7.1: Get the trial download.
- IBM product evaluation versions: Download or explore the online trials in the IBM SOA Sandbox, and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- developerWorks blogs: Check out these blogs and get involved.

About the authors

Raj Rao

Rajesh (Raj) Rao has been working in the area of expert systems and business rule management systems for over 20 years, during which time he has applied business rules technology to build diagnostic, scheduling, qualification, and configuration applications across various domains such as manufacturing, transportation, and finance. He has been with IBM since 2009. With a background in artificial intelligence, his interests include natural language processing and semantic web.

Sandeep Desai

Sandeep Desai is a Senior Certified Enterprise IT Architect with IBM WebSphere's Business Partner Technical Professional team. He works with strategic business partners from startup to large firms. He mentors them to SOA-enable their solution. He evangelizes, educates, and enables partners on IBM software platform. Sandeep is Open Group Distinguished Certified IT Architect, IBM Senior Certified Enterprise IT Architect, and IBM Certified SOA Solution Designer.