<table>
<thead>
<tr>
<th>Page</th>
<th>Title</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>From the Editor</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Semantics Enabling Next Generation SOA - Part I</td>
<td>Johan Kumps, SOASchool Certified SOA Architect - Analyst, RealDolmen</td>
</tr>
<tr>
<td>12</td>
<td>Service Driven SharePoint Enterprise Portal Design</td>
<td>Gijs in ‘t Veld, CTO, motion10</td>
</tr>
<tr>
<td>27</td>
<td>Service-Oriented Design with REST</td>
<td>Raj Balasubramanian, Benjamin Carlyle, Thomas Erl, Cesare Pautasso</td>
</tr>
<tr>
<td>30</td>
<td>Contributors</td>
<td></td>
</tr>
</tbody>
</table>
From the Editor

The new book “SOA with REST: Principles, Patterns, and Constraints for Building Enterprise Solutions with REST” is scheduled to go to print in August, with an official in-print date of August 14th and the official book launch ceremony set for September 24th as part the SOA, Cloud + Service Technology Symposium in London. We’re quite stoked about the release of the new series title as it is the result of some truly deep and interesting research into the convergence of a formal paradigm with a formal architectural model. Many have confused REST as an alternative to SOA, the same way Cloud Computing is sometimes mistakenly viewed as an alternative to Service-Oriented Computing. What we discovered, when delving into the design rules and goals behind service-orientation principles and REST constraints, is that both REST and SOA share a great level of commonality – much more than most expect or assume. There are some points of tension when it comes to certain design qualities, especially those that originate from the external, Web-centric focus of REST versus the internal, enterprise-centric focus of SOA. Overall, however, there are too many benefits to using SOA with REST to ignore. And, regardless of whether it makes sense for an enterprise to do so, it is important to understand the opportunities to leverage the REST architectural style as a legitimate medium for creating enterprise service-oriented solutions.

Thomas Erl, Series Editor and Site Editor
Semantics Enabling Next Generation SOA – Part I
by Johan Kumps, SOASchool Certified SOA Architect - Analyst, RealDolmen

Abstract: This two-part article series discusses how creating and maintaining service based architectures can be a significant challenge and a considerable investment. IT staff must carry out all of the tasks associated with the discovery, composition and invocation of services. Coping with millions of services, solely through human effort isn’t feasible (and I’m not even taking environmental and context changes into account). However, there should be an approach to drive traditional service-oriented architecture to a more dynamic and more flexible pillar in an enterprise architecture through innovative automation.

Introduction

Service-Oriented Architecture (SOA) is a style of software architecture that advocates reusable and intrinsic interoperable units of logic referred to as services. This means that a business application is now just another service composition. In current SOAs the detection and usability analysis of suitable services for a specific client application is limited to manual human intervention.

The human intervention, performed by the project’s architect or analyst, mainly focuses on the selection, negotiation, constraint validation and decision making steps in the business process impacting the flexibility and agility of the solution architecture. Current automation approaches rely heavily on syntactical representations, restricting collaboration, without standardization agreements. Supporting dynamic integration and process automation on an intra- and inter- enterprise level requires addressing the heterogeneity in integration, dynamic constraint validation and runtime agreement negotiations.

In recent years, a lot of research on the Semantic Web, described by Tim Berners Lee has been done. Semantic Web technologies can help overcome the challenges we face today in traditional SOA projects. Realizing service-orientation’s full potential will require integrating SOA and Semantic Web technologies. In this article I will explore how and where semantic web technologies can be adopted to lift traditional SOA to a next level, and form a solid foundation for flexible business solutions and innovative pervasive computing platforms.

The Service Oriented Principle

SOA originally brought eight, nowadays well known, principles and influences to software engineering supported by and applied in a distinct approach to the analysis, design and implementation phases. Within the context of this article, I’ll focus on the Service Discoverability, Service Composability and Service Loose Coupling principles. For each one of them we aim the maximized usage of machine-readable metadata, describing the available information and resources, enabling the automation of a substantial portion of the work related to the application of the SOA principles mentioned above. Let me take the Service Discoverability principle as an example to make my point. Using the currently available tools (UDDI registry/repository, search engines, ...), it is very easy to get lost in a myriad of services, or discover irrelevant ones since only a keyword based search facility is available. For instance, if we search for a service by using the ‘hotel’ keyword we will find all kinds of information mixed with services descriptions. In the table below I’ve listed some of the possibilities:

- Tourist information in the neighborhood of hotels
- Websites or travel agencies providing hotel booking services
- Schools providing courses for hotel personnel
- ...

Copyright © Arcitura Education Inc. 4 www.servicetechmag.com
In the best case we find the wanted service, however in most cases, the precision and recall of the query is low, requiring a human to filter out the usable portion or we don’t find the requested service because none of the available services mention the ‘hotel’ keyword, leading to a less flexible, less dynamic solution.

The Semantic Paradigm

The Semantic Web (Web 3.0) vision was conceived by Tim Berners-Lee defining the Semantic Web as “a web of data that can be processed directly and indirectly by machines”. The main goal of the Web 3.0 is to define a system that enables machines to understand the meaning of the data being provided, searched for, shared and exchanged. Such an understanding requires that the information sources be semantically structured. While the evolution of Web 1.0 to Web 2.0 increased the participation rate of the end-user, by enabling them to interact and collaborate with each other in a social media dialogue as creators of information or content, the transition to the Web 3.0 has more to do with a technology shift. This transition implies that we move from a web of documents to a web of semantically structured data. Web 3.0 gives meaning to the contents of documents and links between them. Content in terms of “entities” such as people, locations, phone numbers and “relationships” between those entities, such as person A, “lives on” address B, “has car” C which is “insured by” insurance company D.

In contrast with documents (PDF, doc, spreadsheets), data can be interpreted by software agents. In order to make sure data can be used by machines, the semantics of this data should be available in a machine understandable manner. This is where Semantic Technologies come into play for knowledge management. These technologies are built on syntaxes which use the Universal Resource Identifier (URI) to represent data in triples-based structures using the Resource Description Framework (RDF). RDF is a framework for representing information about resources in a directed graph formed by triples, expressed using an XML syntax. Each triple (knowledge statement about the real world) is a complete and unique fact made up of subject, predicate and object. The subject is the resource described by the statement. The predicate is the property of the subject identified uniquely by an URI. The object is the value of the subject's property.

Triples can be represented graphically as a directed labeled graph, removing any possibility of confusion over what are subject and objects. By convention the subject is shown as an oval, the predicate as an arc and the object as a box. This convention is depicted in the diagram below, representing the statement: “The book has the title The Semantic Web”.

![Figure 1 – Example of a basic triple](image-url)
A knowledge base describing a specific domain is a collection of RDF triples stored in an ontology. An ontology is thus a formal representation of domain concepts and the relationships between them. Next to capturing domain knowledge in a standard, machine interpretable manner, ontologies can be used to literally reason about the model and data, revealing new insights in the form of additional, not explicitly modeled triples. Subsumption, equivalence and disjointness are some examples of relationships between ontology concepts that could be discovered by the engine. This kind of information is also called inferred knowledge, which is one of the supporting principles of the semantic discovery process discussed later in this article.

To allow standardized description of subjects, objects and the relations between them and other ontological constructs, a RDF Schema (RDFS) was created. RDFS is designed to be a simple data-typing model for RDF. Using RDFS it is possible to model statements like Car is a type of MotorizedVehicle, and that MotorizedVehicle is a sub type of Vehicle.

More detailed ontologies can be created with Web Ontology Language (OWL). OWL is a language derived from description logics, and offers more constructs over RDFS. It is syntactically embedded into RDF, so like RDFS. It provides additional standardized vocabulary (equivalentClass, intersectionOf, inverseOf, ...), useful when modeling domain knowledge.

For querying RDF data as well as RDFS and OWL ontologies, a SQL-like language called Simple Protocol and RDF Query Language (SPARQL) can be used.

When the Semantic Web meets SOA

The foundation of semantically enabled solutions is the domain ontology. The main goal of an ontology is to share a common understanding of the structure of descriptive domain information among people, software agents and between people and information systems, next to enabling reuse of domain knowledge, introducing standards and increasing interoperability. Some examples:

- If I say “house” and you say “maison” how do we know we mean the same thing?
- If I say “vehicle”, how does the system knows if this includes buses, cars, trains, ...?

By making domain assumptions explicit in a standardized manner using ontologies, we achieve interoperability between formerly disparate systems, people speaking different languages as well as the interaction between people and information systems. People can use their own terminology or language to express their needs while the system is able to interpret and link the request to services the system provides.

Regarding Web Services or services in general, semantics can be used to give service components (functional description, inputs, outputs) a meaning. The integration of semantics also allows for the definition of new elements like preconditions and effects (post-conditions), which have not been regarded in syntactic descriptions. Preconditions are logical conditions that need to be fulfilled before the service can be executed. Effects describe changes in the functional context of the service after the service has been executed. Formerly this kind of information was stored in documents made available along with the WSDL contract, but unfortunately not interpretable by software agents. Together with inputs and outputs, preconditions and effects form the service profile.

It is necessary to have a model which can be used as a knowledge base. The most commonly used knowledge base format are ontologies. More specifically semantic service contracts can be modeled using a standard ontology language, OWL-S (formerly DAML-S). OWL-S is an upper ontology for services and will be used to capture semantic information about services in a service inventory. The goal of OWL-S is to support automatic service discovery, invocation, composition and interoperation.
OWL-S models three essential types of knowledge about a service:

- The service profile tells “what the service does” in a way that is suitable for a service-seeking consumer (or software agent acting on behalf of a service-seeking consumer) to determine whether the service meets its needs. The profile models the inputs, outputs, pre-conditions and effect (IOPE’s) of a service. The inputs and outputs in the profile refer to concepts in a published ontology.

- The service model tells a consumer how to use the service, by detailing the semantic content of requests (inputs) and responses (outputs), the pre-conditions of the service, quality of service parameters, the possible effects the service has to its environment.

- A service grounding specifies the details of how a software agent can access the service. Typically a grounding will specify a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. The grounding also specifies unambiguously the way of exchanging data for each input or output specified in the ServiceModel.

![Figure 2 – The top level of the OWL-S ontology](image)

Thanks to the OWL-S Grounding part it is possible to ground a semantic service to a Web Service or to any other type of service (EJB, POJO, ...), maximizing the level of loose coupling between the consumer and the service technology.
Service Discoverability

According to Thomas Erl, the Service Discoverability principle can be defined as follows: “Services are supplemented with communicative metadata by which they can be effectively discovered and interpreted”. In other words service discovery is the process of evaluating a consumer goal or request and returning a set of compatible services capable of fulfilling this goal. First the consumer needs to formulate a service request in order to find appropriate services in the registry. In order for a system to be able to automate the discovery process, the request should define a query based on semantic concepts instead of keywords. This way it is possible to define the goal more precisely and to derive relationships between semantic concepts defined in the request and a service offer. The service query also allows to define pre- and post-conditions for a service.

The service provider is responsible for the service advertisements, describing a service offer with respect to functional capabilities, non-functional aspects, etc. The complete semantic service lifecycle model is depicted in the diagram below:

![Figure 3 – The semantic service lifecycle.](image)

The *matching* component is crucial in the semantic discovery process. It takes the service request and matches it with the available service advertisements. The actual matching is a pairwise comparison of a service advertisement and a service request. More specifically, the matching engine calculates the similarity between the concepts in the service request and these used in the service advertisements using an ontology reasoner, like the Jena framework (http://jena.apache.org/).
The OWL-S service profile contains enough information for a matchmaker to determine whether a service satisfies the requirements of a consumer. Several matchmaking algorithms rely on the matching of inputs and outputs of the service profiles. Four matching degrees can be identified:

- **Exact**
  In case an output of an advertised service is an equivalent concept to a requested output or if the advertised output is a superclass of the requested output, an exact match is considered between these concepts.

- **Plugin**
  If an advertised output subsumes a requested output, the relation between these concepts is weaker as compared to the exact match since subsumption is inferred by the reasoner. Consequently the matchmaker component will infer whether the advertised output can be plugged in place of the required output.

- **Subsume**
  The advertised output is a subset of the requested output.

- **Fail**
  We consider to have a fail match between concepts in case none of the above conditions are true.

These degrees are ranked as follows:

**Exact > Plugin > Subsumes > Fail**

This means that exact is a more desirable match than plugin, etc.

The combination of four methods leads to a more fine-grained ranking scheme though. We identify the profile hierarchy, the input/output types, preconditions/effects and QoS constraints matching. Two of them, profile hierarchy and input/output matching are based on the above mentioned matching degree calculation.

Within the profile hierarchy matching phase all advertised services presenting a profile that at least subsumes the concepts expressed in the consumer request are considered as services potentially fulfilling the consumer’s goal.

The input/output parameter matching searches for services that produce an output required as an input of a given service. A matching service is a service that can correctly perform a task if all input concepts defined in the advertisement are satisfied by the consumer and if all output concepts defined in the request are satisfied by the advertisement. Next to exact matching, when an output type is more specific (subsumes) than a required input type, matching is also considered successful.

A precondition of a service specifies a condition that needs to be satisfied before the service logic can be performed successfully. Contrasting with preconditions, post-conditions or effects specifies conditions that hold as a result of a successful execution of the service. The matching engine looks for services that result in an effect that can fulfill the preconditions mentioned in the service request. A service is considered a match if all preconditions defined in the advertisement are satisfied by the consumer, in other words, a service can fulfill a consumer’s goal if all effects defined in the request are satisfied by the advertisement.

Next to the above mentioned data (input/output) and functional (preconditions/effects) semantics we can identify non-functional semantics referring to quality of service (QoS) or general requirements and contraints too. QoS constraints matching will be discussed within the context of a service composition later in this article.

Let us now look at an example of how a request is matched with advertised services using profile hierarchy and IOPE’s matching. Suppose that the user’s requirement is “to book a commercial flight from Brussels to London on 23rd of september 2012”. The profile hierarchy matching engine searches for a subsume of the
CommercialFlightProduct. Based domain knowledge base the concept FlightBookingService is obtained. The system then considers services that are instances of this concept, i.e. RyanAirFlightBookingService, BritishAirwaysFlightBookingService and the BrusselsAirlinesFlightBookingService.

Consequently the engine will perform an input/output matching. All services that provide an output that matches a required input are retrieved. Based on the domain knowledge base the engine determines that only the BritishAirwaysFlightBookingService can perform the booking using the given departureCity (Brussels), the arrivalCity (London) on the departureDate (23rd September 2012). The third phase is the evaluation of pre- and post-conditions. The engine searches for services that result in an effect that can fulfill the requested precondition.

Both the RyanAirFlightBooking and BrusselsAirlinesFlightBooking services request a flight number and a departure date as input parameters. Due to the fact that the consumer is not able to provide a flight number, neither of these services can fulfill the consumer’s goal on their own. This is where service composition comes into play which is discussed in the next paragraph.

Conclusion

The second article in this two-part series goes over Service Composability, QoS – awareness, and the conclusion to this article.

Acknowledgements

I wish to thank my colleagues Jan De Bo and Steve Van Den Buys who reviewed and critiqued my article.
Arcitura Education Inc. has announced a new certification for the Cloud Certified Professional (CCP) program from CloudSchool.com, dedicated to cloud computing-based virtualization technology and practices. The new Certified Cloud Virtualization Specialist designation requires the completion of Prometric exams C90.01, C90.02, C90.16, C90.17 and C90.18. This certification track correspondingly introduces three new CCP courses and self-study kits.

**CCP Module 16: Fundamental Cloud Virtualization**
Core topic areas pertaining to the fundamental virtualization mechanisms and types used within contemporary cloud computing platforms.

**CCP Module 17: Advanced Cloud Virtualization**
A range of specialized and advanced topics that build upon module 16 to explore virtualization-related reliability, performance and integration, as well as combinations of mechanisms.

**CCP Module 18: Cloud Virtualization Lab**
A hands-on lab during which participants apply the models, concepts, and techniques covered in previous courses, in order to complete a series of complex exercises.

**Workshops & Self-Study**
Attend an instructor-led workshop or purchase the official Cloud Virtualization Specialist Certification Self-Study Kit Bundle.
Service Driven SharePoint Enterprise Portal Design

by Gijs in ’t Veld, CTO, motion10

Abstract: This article helps understand how a service driven enterprise portal design with SharePoint as the front-end and Microsoft’s SOA based integration middleware underneath enables the use of composite services to help build more efficient, scalable and supportable central workspaces that can be used in modern BYOD environments.

Introduction

Microsoft’s SharePoint is a pretty dominant collaboration platform, serving more than 70% of enterprises world-wide. Besides enabling more efficient sharing of documents, improving findability and providing the functionalities to collaborate with team members on projects and documents, SharePoint can also be implemented as an enterprise portal. Basically, by providing document management features, intranet capabilities and integration with other systems you can implement a true central workspace for employees, customers and partners providing them with self-service capabilities.

Of course SharePoint is well known for natively storing and sharing of all kinds of documents and files, as well as allowing people to collaborate with each other on these documents. However, in most SharePoint solutions the information for a great part also comes from external systems. This is why with SharePoint implementation projects integration can often play the greater role.

The integration with other systems and services should be organized according to the SOA principles, thereby providing the means to incorporate external data and processes in the portal, and creating an environment where everything comes together in the right context and with the right security trimming. By implementing a SOA based service layer underneath SharePoint, building such solutions will become much easier, and the end result will be much more scalable and supportable.

The Classic Trap

When leveraging SharePoint as an enterprise portal, it is important to know that it is quite easy to fall into the classic trap that Microsoft tooling is “famous” for: it is quite easy to get to know the product quickly and build your first “hello, world” applications. However, before you know it you will end up with all kinds of functionality that is being used in production environments, and the company is quickly faced with an intranet or portal solution that is not easily manageable, scalable, stable, or even worse an environment that is not supportable.

SharePoint is not a simple product. It can do many things, and inherently its architecture is quite complex. Before designing and building SharePoint solutions, it is therefore crucial that you know the architecture very well, and you know all the ins- and outs of the platform with regard to extensibility. It is very easy to blame the platform on which your badly designed solutions have been built, and before you know it, SharePoint has a bad name within your company, which is just not fair.

In order to prevent such situations, building solutions using SharePoint as the platform have to adhere to a number of simple rules:

1. Before you start building “custom solutions” make sure it is not (more-or-less) available out-of-the-box already; discuss your requirements with a SharePoint consultant

2. Make sure you build solutions according to the SharePoint architecture; discuss your solution design with a SharePoint architect and have it checked against the Project Start Architecture
1. Check your solution design against the SharePoint Governance Plan; this will make sure that the design adheres to the guidelines and rules created for your company’s intranet or portal.

These rules apply to SharePoint solution design and development in general. Specifically geared towards integration of external content there are two more rules:

1. Make sure to apply the principles of service design [REF-1] when exposing data and processes as services in your SOA.
2. Implement a service layer to streamline the integration of external content and make it manageable and supportable.

The remaining part of this article will discuss the integration of external data within SharePoint in more detail.

**Business Connectivity Services**

Out-of-the-box, SharePoint comes with features to integrate information from other applications. In the latest version this is called “Business Connectivity Services” (BCS). While using this technology, it is possible for “External Content Types” (ECT) to be used just like any other content type in your lists and libraries. But instead of retrieving the information from native data stores, SharePoint will use BCS to retrieve the data. An ECT defines the schema and operations to perform on the data.

![SharePoint Business Connectivity Services (BCS) architecture](image-url)
As can be seen in figure 1, BCS can retrieve and write data from and to other sources by either using direct SQL queries by invoking .Net components or by consuming Web Services. Through BCS you have full CRUD functionality. There are however several limitations, the lack of functionality to compose services, and the use of data format conversion are the most important ones here. It is not (easily) possible to combine several data sources into one, and display that in a standard SharePoint list or webpart.

Furthermore, it is quite easy to fall into yet another trap; this time it’s the “spaghetti integration” trap, where many systems and services communicate with each other in a peer-to-peer fashion and thereby create an unmanageable situation.

When you take a look at the online BCS samples, most of them show you how easy it is to connect to SQL Server tables. Nice indeed, but what if you are not the owner of this SQL database? What if an application update changes column or table names? Are you going to update all ECTs you have defined previously? Even if you can (think about the trouble a table split would cause you), it might incur a lot of rework in all sorts of places. Another example is on WCF (web services) connections: What if your products are now stored in two warehouse locations while they were previously stored in one? You would need to rewrite your BCS connection and change it to a custom .Net Assembly connector.

By implementing a service layer underneath SharePoint BCS, it becomes possible to consume composite services and implement service versioning, real end-to-end tracking and tracing, and monitor features. Also, implementing a service layer gives more flexibility and capabilities with regard to the handling of transactions and providing scalable, failsafe solutions.

And maybe most importantly, BCS in combination with a service layer and the claims based security features provided by SharePoint will create a secure environment, where information will be handled according to the security regulations, and the roles and responsibilities defined in the SharePoint Governance Plan.

Implementing a Service Layer

There are several ways to implement a service layer underneath your SharePoint enterprise portal of which SharePoint Service Applications and BizTalk Server (on-premise) or Azure Service Bus (cloud) are the most commonly used ones in “Microsoft unless” environments. BizTalk Server is used most often because typically implementing a SOA architecture is done through the use of an ESB, and BizTalk Server can be implemented according to the ESB design pattern. If SharePoint runs in the cloud (SharePoint Online), Azure Service Bus (implemented according to the ESB design pattern) is a better choice. For hybrid solutions, where part of the services run on-premise and part in the cloud, the hybrid Federated Service Bus pattern would be the best solution (see my previous article in the April 2012 edition of this magazine [REF-3]).

SharePoint then just leverages the (Federated) ESB for consuming and exposing services. For simple, SharePoint only scenarios though, Service Applications could suffice. The remaining part of this article however focuses on the implementation of an ESB as a service layer.

External Contents Types in SharePoint can be organized and stored in the External Content Type Repository (see figure 2). For example, the ECT “Customer” can be used by a SharePoint power user to display “live” customer data in a webpart on a page that contains all previously exchanged documents for that particular customer. It is also possible to use several different webparts on the same page that through the use of the connected webparts technology can share key data and display information in the right context. For example, showing the “live” customer data, and showing all the relevant RFQ’s and Quotes for that customer in other sections on the page is an easy task by just sharing the “customer id” between the webparts.
The key is that these ECT’s are directly bound to services provided on the underlying ESB. When the Project Site home page for “Project X for customer Y” is displayed, it will retrieve the live Customer, RFQ and Quote data for Customer Y through the ESB, which in its turn gets the data from the underlying back-end systems and services through (orchestrated) web services calls.

It gets more interesting if service composition is needed. For instance, when providing an order interface for your customers in the extranet section of your portal. The “Order” ECT would then consume the Order service on the ESB, which could be implemented as a Scatter Gather pattern like depicted in figure 3. The bus exposes the Order service as a web service consumed by SharePoint through the ECT. The Order service is implemented as a composite service that invokes several other services based on the needs. For example, it could invoke a service provided by a CRM system to check if the customer is a known customer and provide a reference to it. The next step could be a credit check provided by a cloud service. A final step could be submitting the customer’s order to the ERP system. The results of all the service calls are aggregated and returned to the SharePoint ECT, which will use it to display the results of the order entry in the portal.
Business processes can be modeled within the ESB, which can combine the various backend systems’ functionalities, and is able to expose them as composite services for consumption, by SharePoint. The business process that runs in the ESB is responsible for orchestrating the interaction with all underlying services, and handling this process as a single transaction that can be rolled back if necessary.

![Figure 3 – Scatter Gather pattern through the ESB](image)

The ESB makes use of its own services to fulfill the requests, such as:

- Itinerary web service: Determine which itinerary (or routing slip) to apply to the request
- Routing service: Routing the request to the different endpoints
- Protocol adaptation service: Make it possible to invoke services that are not based on standard web services
- Transform service: Transformation between requests and canonical data model and canonical data model and service providers
- Process orchestration: For more elaborate orchestration (beyond itinerary actions) of processes, including the use of the business rules engine to make decisions

By implementing a full featured ESB like BizTalk Server or Azure Service Bus underneath SharePoint, the following extra advantages will become available:
- Service location and version transparency
- Transport protocol conversion
- Error handling and repair
- Data format transformation
- Message interactions
- End to end tracking and tracing
- (Business) activity monitoring

The last one is a particularly interesting one. Because all the (composite) services handled by the ESB are monitored already, you can easily extend the information being monitored to the level that Business Activity Monitoring becomes possible. By tracking milestones in your business processes or even tracking certain business content retrieved from the transactions processed by the ESB, you can feed your Business Intelligence solution (for example through SQL Server and SharePoint BI capabilities) and use the information to fine-tune your business processes. A true Business Process Management (BPM) implementation is not farfetched anymore then.

Performance, Latency and Mobile Access

Performance and latency is always an issue with any integration solution or pattern. Most of the times, the performance of the underlying services is the bottleneck. But in order to make the ESB itself perform as quickly as possible and to be able to handle large volumes of service calls and transactions, a number of things can be done. First, it is important to notice that BCS does not support caching (apart from metadata caching). Second, it is important to know that in an ESB that provides guaranteed delivery, transaction handling and restart ability. It is evident that data and milestones are persisted to a database. Therefore, two things are absolutely crucial to implement in order to have a great performing ESB underneath your SharePoint:

1. A dedicated, high performance and scalable database
2. A forward cache

The 2nd point is a typical solution to provide for high performance. Low latency service call replies in environments where lots of the same data is queried over and over again, such as customer data, employee data, etc. This pattern is described in Enterprise Integration Patterns [REF-2].

By instrumenting your ESB correctly by means of performance counters, correct milestones and other metering information in your activity monitoring layer, it is always possible to at any given time determine where bottlenecks are in your integrated application landscape and adjust accordingly.

On a last note, SharePoint portals can be implemented and extended as such that touch-friendly mobile access is possible as well. There are several technologies available for developing such solutions, such as jQuery Mobile. A lot of times, SharePoint is implemented in such a way that you do not even know SharePoint is the platform underneath all the user interaction. For example, lots of SharePoint implementations use the Office Applications as the end-user interface, such as Outlook. A human workflow started from within a process orchestration in the ESB could easily create a task in Outlook through a SharePoint workflow. Mobile access to your Outlook or other Office applications then automatically takes care of the rest.
Conclusion

While implementing a SharePoint enterprise portal with the above guidelines in mind, a true any time, any place, any device environment can be created that will deliver the right information in the right context at the right time to employees, customers and partners.

Leveraging SharePoint as an enterprise portal requires extensive knowledge of the SharePoint platform architecture and service-oriented architecture. By building and exposing services according to the principles of service design and leveraging the services provided by an ESB (either on-premise or in the cloud, or in a hybrid fashion using the Federated Service Bus design), any SharePoint enterprise portal becomes a much more scalable, supportable and sustainable integrated solution.

Implementing SharePoint on top of a SOA based service bus makes it possible to design true BPM solutions that can help companies quickly adapt their business processes to the employee’s and customer’s needs.

References

Cloud Architect Certification

Self-Study Kit Bundles
Now Available for Pre-Order
with Special Pricing

www.cloudschool.com/certifications/architect
Lessons Learned: Best Practices for a Successful Introduction of Business Process Management (BPM)

by Kai Wähner, IT Consultant, MaibornWolff et al

Abstract: Business Process Management (BPM) is complex, expensive, and often fails! If you agree (in the year of 2012+), then you should read the following rules to do BPM correctly in your next project.

Introduction

This article does not give an introduction to BPM. It starts with a use case immediately to show best practices and common miscues regarding BPM. If you are not familiar with BPM, BPMN, WS-BPEL or related topics, then you should begin with a BPM introduction [REF-1].

Use Case: Loan Request

Figure 1 shows a stateful, long-running business process, which is applied in each paragraph to explain best practices and possible miscues in BPM projects. Here are the steps in this process: First, the business process covers a loan request. After the loan request is received, some automatic scripts and service tasks are executed. Following that, user interaction is required to approve or reject the loan request. Depending on the evaluation, the request is processed or denied. Stakeholders here have either the job role of a business expert or a developer.

Looking at this use case, let’s discuss several different aspects of BPM to learn important best practices and miscues in real world projects.
Goals of BPM

BPM attempts to improve processes continuously. It can therefore be described as a “process optimization process”. Business-IT-Alignment is the key to improve processes continuously. The important word here is “alignment”. It is not just about IT! It is not just about business! It is about working together - one of the major challenges of many BPM project. Besides this, BPM has further goals (which can be reached as consequence of good Business-IT-Alignment):

- Increased efficiency
- Higher transparency
- Better quality
- Reduced costs
- Enabling new business models

Use Case: The primary goal is to reach higher transparency and improve communication between business and IT. When introducing BPM, the focus should be on this goal right from the very beginning. The goal has to be supported and made transparent by decision makers. It would be wrong to introduce BPM without any key goals. In addition to this, key performance indicators should also be defined to measure the success of BPM.

Rule 1: If you want to do BPM correctly, then be aware that your primary goal is to improve Business-IT-Alignment!

BPM and SOA

BPM is an offspring of Service-oriented Architecture (SOA) as Anne Thomas Manes explained in her well-known article “SOA is Dead” in 2009 [REF-2]. SOA is the foundation for modern approaches such as BPM, cloud computing or mashups. So, “BPM based on SOA is technology’s response to the growing demand for a flexible business environment that is not hindered by application silos. The integration technology must loosely couple the applications and resources that make up the process, otherwise the logic of a process will get hard coded into a particular technology platform, which may be expensive to change and therefore defeat the entire purpose of BPM.” [REF-3]

Many risks have to be considered when pursuing BPM without SOA because advantages of BPM cannot be gained without services [REF-4].

Use Case: The loan request is already split up into several loosely coupled services (script and Java service tasks, human interactions). Therefore, it is ready to be realized with BPM. If everything is regarded as one silo, BPM would not be possible, and higher flexibility and increased efficiency could not be reached.

Rule 2: If you want to do BPM correctly, then do not try to realize it without a service-oriented Architecture!

BPM Lifecycle

Let’s first tell you what BPM is not: BPM is not just tooling. Instead, BPM is an engineering process with several different stakeholders, technologies and tools. The lifecycle of BPM is iterative and has the following phases:
■ Design
■ Modeling
■ Execution
■ Monitoring
■ Optimization
■ Start again: Design
■ ...

Of course, tooling should support the whole lifecycle. Therefore, BPM tools should be evaluated after some processes are modeled and designed, and when it is clear how business processes shall be executed and monitored.

It is important to introduce BPM iteratively. Do not try to “boil the ocean by implementing a BPM / SOA solution for an entire business group or division as one massive upgrade or roll out. BPM and SOA efforts are both inherently evolutionary in that they are best implemented in small, constrained, and frequent capability releases that grow upon each other.” [REF-5]

**Use Case:** First choosing a tool and then starting realization is the wrong approach, and can lead quickly to failure. Several requirements may be unknown at the beginning of the project. For instance, no complex business rules have to be defined. So therefore, a BPM tool that integrates a complex (and expensive) rules engine would be a very bad decision. Following the BPM lifecycle iteratively reduces efforts, helps analyzing existing services and roles, and enables reusing available knowledge and existing source code.

**Rule 3:** If you want to do BPM correctly, then be aware that BPM is not just tooling, but an engineering process with a huge, iterative lifecycle!

**Benefits of BPM**

BPM is no Swiss army knife, which can solve every problem. Do not use BPM in wrong situations. I don’t like seeing people using BPM tools just because there is a workflow. You don’t need a BPM tool to realize a workflow. In this case, you can use (for instance) a Java method that does some business logic, and then calls another Java method that does some business logic, and then calls another Java method, and so on...

BPM brings benefits especially if you need:

■ Long-running stateful workflows
■ Frequently changing processes
■ Human interaction

**Use Case:** BPM is an excellent choice due to long-running stateful workflows and human interaction. Several benefits are gained, e.g. better transparency, and higher flexibility. Introducing BPM pays off quickly. Implementing such process logic by oneself is a lot of effort, error-prone, and just unnecessary.
Rule 4: If you want to do BPM correctly, then use it only when you really gain benefits!

Myths of BPM

John Cingari has defined four myths of BPM projects, which are true in most organizations [REF-6]:

1. Business analysts WILL create executable process models
2. Business analysts CAN create executable process models
3. Business analysts WANT to create executable process models
4. IT WANTS business analysts to create executable process models

There will always be business guys and IT guys, and they have a different understanding of things. Business guys will never be able to model complex business processes. Therefore, Business-IT-Alignment is very important, and tooling cannot solve this problem. Business and IT have to work together to do BPM successfully.

Use Case: Business analysts have a lot of knowledge about loan request procedure. If possible, let them model the processes. Developers just add technical details. However, many business people cannot think this way, or simply do not want to. On the other hand, developers learned this way of thinking and modeling from the beginning of their carrier. It is fine if developers model all business processes. It is much more important to pick everybody up from business. Hold meetings to gather information, then model processes, and then do review meetings. Iterate this process.

Rule 5: If you want to do BPM correctly, then do NOT believe in BPM myths!

BPM Standards

Standards are important for BPM to be future-proof and vendor-independent. This way, different tools can be used for different phases / by different people. A lot of standards exist for BPM, such as BPMN, XPDL, BPEL, jPDL, WF-XML, ARIS EPC, and others.

The most important standard for BPM is BPMN [REF-7] (Business Process Model and Notation) - at least since version 2.0 (released in March 2011). BPMN 2.0 added a standardized XML description format. Now, BPMN is not just flow charts: Sufficient restrictions and constraints specify the execution of business processes non-ambiguously. Another important new addition to BPMN 2.0 is its standardized extension points (because every vendor always adds his own specific features to a product). For instance, the open source BPM framework Activiti offers an extension for a Java service task.

WSBPEL [REF-8] (Web Services Business Process Execution Language, short: BPEL) is the second most important BPM standard. However, BPEL can only be used for SOAP Web Service execution. Due to the fact that BPMN 2.0 also includes execution of business processes, new products do not need to use BPEL for execution. Nevertheless, a BPMN 2.0 implementation must align to BPEL to be 100% standard conform. If you ask yourself about is the relationship between BPMN 2.0 and BPEL 2.0 and if BPMN 2.0 makes BPEL 2.0 redundant, then you should read the interesting interview at [REF-9].

Use Case: BPMN is the defacto standard for BPM in 2012+. It is suitable for business and IT people, and it can be used for modeling and execution. Many other standards are outdated or way too complex. Choosing WS-BPEL instead of BPM would add several restrictions and increase complexity. If the available tool in your organization just supports WS-BPEL, you can still use BPMN for modeling and WS-BPEL for execution.
Rule 6: If you want to do BPM correctly, then use BPMN 2.0!

**BPM Tools**

Countless BPM products are available on the market. How do you decide which one to choose? You can do an evaluation comparing several criteria, e.g.:

- Open Source vs. Proprietary
- Supported Technologies
- Supported Standards
- GUI Designer
- Community, Documentation, Commercial Support
- ... and so on

Right?

Wrong! This procedure might work for most product or framework comparisons. I often created such criteria lists before by myself. However, for BPM tools, there is one important criterion, which must be discussed before you look at all the other ones: Do you want to use a developer-focused tool or a designer-focused tool?

Why are these so different?

A developer-focused tool such as Activiti or JBoss jBPM is embedded directly into your container or standalone application. You use your common development tooling and environment. You write source code including corresponding unit tests. Usually, developer-focused tools are very lightweight, easy-to-use, open source and offer an API for your preferred language (e.g. Activiti is Java-based). You can start using it after some minutes. However, developer-focused open source tools are not perfect. For instance, they often provide bad documentation and a very rudimentary web frontend for modeling and monitoring.

On the other hand, there are designer-focused tools, which promise zero-coding. These tools are real products, not just embeddable frameworks. You do not write source code. You create business processes in a graphical designer via drag and drop. No coding is required. Decide for yourself if this is a pro or con. Sometimes, zero-coding might be a con (remember the four myths about BPM - business analyst cannot model business processes). If you cannot code anything or if you have to use ugly workarounds, then this can be a show-stopper for real world projects, where the tool does not offer the feature, which you require.

Proprietary, designer-focused tools are very expensive and heavyweight solutions (e.g. you do not need to buy and install just Oracle BPEL Suite, but also Oracle WebLogic Application Server, and several other products from the Oracle stack). If you start installing such a suite on day 1, you probably will not be able to use it until at least day 2 (because installation is so complex and time-consuming). Contrary to developer-focused open source tools, proprietary products offer high stability, powerful web frontends and good integration into other parts of your development environment (if you use the whole stack from one vendor).

Besides heavyweight, proprietary BPM tools from large vendors such as Oracle or IBM, there is also some designer-focused open source tools available, e.g. ProcessMaker or Bonita Open Solution. They are easy to setup and learn. Usually, designer-focused open source tools are not as powerful as proprietary products. However, you can add required functionality by yourself because they are open source.
So, the first question to answer for BPM tool choice is: Do I need / want a developer-focused tool or a designer-focused tool? After you have made this decision, you can evaluate BPM tools by other criteria: Which features are supported (e.g. a complex rules engine)? Is the tool extensible, can other software be integrated easily? Which technologies are supported? How shall business processes be monitored? And so on...

**Use Case**: Process logic of the loan request contains only business information. No complex logics or flows are needed. Technical information is not important on this level. Therefore, business analysts are able to model this process. Afterwards, developers just add technical interfaces for execution. This use case is a perfect match for a lightweight, designer-focused product.

**Rule 7: If you want to do BPM correctly, then choose the right BPM tool for the job!**

**BPM and Systems Integration**

You can use a BPM tool to integrate systems. For instance, there is a standard BPMN service task for SOAP Web Services. So, you can use whatever technology you want. Besides, you can use script service tasks (for e.g. Groovy or JavaScript), Java service tasks (as extension), and others.

But is this a good solution? It may be fine, if you just have to call one or two services in your process. In all other situations, the magic word(s) is “separation of concerns” Otherwise you will end up in complex spaghetti solutions. This is why a SOA is so important for introducing BPM.

A BPM tool is not an integration tool, but a process engine (plus some addons, of course). Use BPM for its benefits, e.g. realizing long running stateful processes or human interactions. For systems integration, use a tool that is built for this job, i.e. a lightweight integration framework such as Apache Camel, or an Enterprise Service Bus (ESB) such as IBM WebSphere Message Broker. Wolfgang Pleus wrote a great (short) blog post about combining a BPM engine and an integration framework [REF-10].

**Use Case**: Some technical services have to be called from the loan request business process. Deep technical knowledge for routing, transformation, and connectivity to different systems is required. Therefore, this logic should be separated from the process logic. The business process just contains the interfaces. All other integration logic is realized with an integration solution. Effort for separation of concerns is low due to an flexible, loosely coupled SOA fundament.

**Rule 8: If you want to do BPM correctly, then do NOT use it for systems integration!**

**References:**


[REF-10] Combination of a BPM Engine and an Integration Framework http://www.pleus.net/blog/?p=1028
SOA with REST
Principles, Patterns & Constraints for Building Enterprise Solutions with REST

"An inspirational book that provides deep insight into the design and development of next-generation service-oriented systems based on the use of REST. This book clarifies the convergence of SOA and REST with no-nonsense content that addresses common questions and issues head-on. An essential 'instrument of modern service implementation' and a powerful body of knowledge for software designers, architects and consultants."

Pethuru Raj PhD,
 Enterprise Architecture Consultant,
Wipro Consulting Services

"This book illuminates the connection of the two domains - SOA and REST - in a manner that is concrete and practical, providing concise application to every day architectural challenges. Fantastic!"

Ryan Frazier,
Technology Strategist,
Microsoft Corp.

The soon to be released “SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST” book has received an official in-print date of August 3, 2012. This is the eighth published book, as part of the recently rebranded, Prentice Hall Service Technology Series. The book authored by Thomas Erl, Benjamin Carlyle, Cesare Pautasso and Raj Balasubramanian will officially be launched the following month at the 5th International SOA, Cloud + Service Technology Symposium.

To learn more about this book, visit: www.servicetechbooks.com/rest

TABLE OF CONTENTS

Chapter 1. Introduction
Chapter 2. Case Study Background

PART I: FUNDAMENTALS
Chapter 3. Introduction to Services
Chapter 4. SOA Terminology and Concepts
Chapter 5. REST Design Constraints and Goals

PART II: RESTFUL SERVICE-ORIENTATION
Chapter 6. Service Contracts with REST
Chapter 7. Service-Orientation and REST

PART III: SERVICE-ORIENTED ANALYSIS AND DESIGN WITH REST
Chapter 8. Mainstream SOA Methodology
Chapter 9. Analysis and Service Modeling with REST
Chapter 10. Service-Oriented Design with REST

PART IV: SERVICE COMPOSITION WITH REST
Chapter 11. Fundamental Service Composition with REST
Chapter 12. Advanced Service Composition with REST
Chapter 13. Service Composition with REST Case Study

PART V: SUPPLEMENTAL
Chapter 14: Design Patterns for SOA with REST
Chapter 15: Service Versioning with REST
Chapter 16: Uniform Contract Profiles

APPENDICES
A. Case Study Conclusion
B. Industry Standards Supporting the Web
C. REST Constraints Reference
D. Service-Orientation Principles Reference
E. SOA Design Patterns Reference
F. State Concepts and Types
G. The Annotated SOA Manifesto
H. Additional Resources
Service-Oriented Design with REST
by Raj Balasubramanian, Benjamin Carlyle, Thomas Erl, Cesare Pautasso

Using the conceptual service candidates modeled during the preceding service-oriented analysis process as a starting point, service-oriented design is dedicated to the physical design of service contracts. When it comes to contract design with REST, we need to be concerned with two particular areas: 1. The design of a uniform contract for a service inventory and 2. The design of individual service contracts within the service inventory and in compliance with the uniform contract. The uniform contract needs to be firmly established before we begin creating service contracts that will be required to form dependencies on uniform contract features. As a service inventory grows and evolves, new services can still influence the design of a uniform contract, but uniform contract features are generally changed and added at a very deliberate pace.

Following the preceding sequence, this sample chapter from the new book “SOA with REST: Principles, Patterns, and Constraints for Building Enterprise Solutions with REST” begins with coverage of uniform contract design topics and then moves on to topics that pertain to the design of REST service contracts. The chapter concludes with a section on complex methods, an optional field of REST contract design and one suitable mainly for use within controlled environments, such as internal service inventories.

Read sample chapters at: http://servicetechbooks.com/rest/sample_chapters
Certified Cloud Architect
October 1-5, 2012
Las Vegas, NV, USA

Certified SOA Architect
October 8-12, 2012
Utrecht, Netherlands

Cloud Technology Professional
October 15-17, 2012
Las Vegas, NV, USA

Cloud Technology Professional
October 15-17, 2012
Sydney, Australia

Certified SOA Consultant
October 15-19, 2012
Vancouver, BC, Canada

Certified SOA Architect
October 22-26, 2012
Fairfax, VA, USA

Certified Cloud Architect
October 22-26, 2012
Munich, Germany
Johan Kumps

Johan Kumps is a SOASchool Certified SOA Architect - Analyst at RealDolmen, a Belgian consultancy organisation based near Brussels. For several years Johan has been involved in the service oriented analysis and design phase of several Service Oriented Architectures for both Flemish and Belgian Federal Government.

In this statute Johan designed the Federal Service Bus (FSB) platform for the Belgian Federal Government using Oracle technology. The FSB is an implementation of the Enterprise Service Bus pattern enabling different government divisions to exchange information using SOA concepts. Currently Johan is fulfilling the roles of SOA Architect and SOA Analyst in several SOA projects. Thanks to his everlasting interest in new things and taking what he’s doing to the next level, Johan came across Semantic Web technologies a few years ago. Some time ago he realized that this new approach to the Web was becoming more and more mature. Johan started his own investigation and rapidly came across some colleagues that had already been investing time in this matter. No further convincing was necessary to make Johan see that he needed to sink his teeth in the combination of semantic technologies and service oriented principles. As a committee member of the JBoss ESB project, he started with the implementation of Semantic SOA features in the JBoss ESB platform.

Contributions

■ Semantics Enabling Next Generation SOA – Part I
■ The Dutch-Translated SOA Manifesto & Annotated SOA Manifesto

Gijs in ’t Veld

Gijs in ’t Veld is CTO at motion10, where he is specialized in Enterprise Integration and Portals. Since 2001 he has been involved with hundreds of integration projects world-wide on the Microsoft Business Platform. Gijs has been awarded Microsoft MVP 7 times in a row for his many contributions to the world-wide adoption of BizTalk Server and Windows Azure in the past decade and has been a Microsoft V-TSP (technical sales professional) Integration since 2007. He is also a technology advisor for the Microsoft Business Platform technologies Windows Azure AppFabric and BizTalk Server.

Contributions

■ Service Driven SharePoint Enterprise Portal Design
■ Guidance for Integration Architecture on the Microsoft Business Platform
Kai Wähner works as an IT-Consultant at MaibornWolff et al in Munich, Germany. His main area of expertise lies within the fields of Java EE, SOA, Cloud Computing, and Enterprise Architecture Management. He is speaker at international IT conferences such as JavaOne or Jazoon, writes articles for professional journals, and shares his experiences with new technologies on his blog (www.kai-waehner.de/blog). Contact: kai.waehner@mwea.de or Twitter: @KaiWaehner.

Contributions

- Lessons Learned: Best Practices for a Successful Introduction of Business Process Management (BPM)
Join the thousands of members of the growing international Arcitura community. Launched for the first time in mid-2011, Arcitura Education made official social media communities available via LinkedIn, Twitter, and Facebook. These new communities join the already existing memberships of LinkedIn, Twitter, and Facebook platforms for the Prentice Hall Service-Oriented Computing Series from Thomas Erl and the International SOA, Cloud + Service Technology Symposium Series.
The Service Technology Magazine is a monthly online publication provided by Arcitura Education Inc. and officially associated with the "Prentice Hall Service-Oriented Computing Series from Thomas Erl." The Service Technology Magazine is dedicated to publishing specialized articles, case studies, and papers by industry experts and professionals in the fields of service-oriented architecture (SOA), cloud computing, semantic Web technologies, and other areas of services-based technology, innovation, and practice.

www.servicetechnologymagazine.com
www.servicetechmag.com