Componentization of Business Process Layer in the SOA Reference Architecture

Liang-Jie Zhang [1], Jia Zhang [2]
[1]IBM T.J. Watson Research Center, USA
[2]Department of Computer Science, Northern Illinois University, USA
[1] zhanglj@us.ibm.com,[2] jiazhang@cs.niu.edu

Abstract

This paper discusses and analyzes the Business Process layer in the SOA Solution Stack (S3) model, which is also known as the SOA Reference Architecture (SOA-RA). Business Process layer leverages the Service layer to quickly compose and choreograph services and to coordinate business processes to fulfill customer requirements. Based on industry practice, we introduce a set of architectural building blocks of the layer, together with the interdependencies and interactions between them, to componentize the Business Process layer in the context of SOA Reference Architecture. We also report industry experiences of applying this layer in SOA solution engagements.

Keywords: Solution modeling, S3, Business Process layer, SOA Reference Architecture, SOA.

1. Introduction

Service Oriented Architecture (SOA) is well acknowledged by its power of integrating and composing existing software components into new business processes or applications [1]. Business applications designed and developed in such a way are called SOA solutions or services solutions. Based on industry practice, IBM proposes Service-Oriented Solution Stack (S3) as the SOA Reference Architecture (SOA-RA) to guide IT architects in designing the overall architecture of an enterprise-level SOA solution [2]. S3 is also known as SOA Solution Stack. Nine layers are identified: Operational System layer, Services Component layer, Service layer, Business Process layer, Service Consumer layer, Integration layer, Data Architecture layer, QoS layer, and Governance layer.

S3 alone is coarse grained and does not provide normative guidance for designing Architectural Building Blocks (ABBs) in large-scale SOA solution design and development. For example, we introduced a set of ABBs as fundamental configurable and reusable units for Consumer layer [3] in the S3 model. An ABB is an autonomous component that encapsulates internal states and functions and can be configured and extended. Each layer in S3 is comprised of a set of collaborative ABBs; so that solution architects can configure and customize the ABBs and rapidly prototype their application-specific SOA solutions.

In addition, we introduced ABBs in the QoS layer and Data Architecture layer [1] and Service Component layer [4]. In [5], we present a theoretical framework to support and facilitate the design and development of ABB-based SOA solutions. This paper reports our continuous work and introduces a set of industry practice-based ABBs, as well as the interdependencies and interactions between them, for the Business Process layer. Business Process layer leverages the Service layer to quickly compose and choreograph services and to coordinate business processes to fulfill customer requirements. Note that practitioners may define their own ABBs or ABB instances for their own SOA models, based on their own best practice. We also report our experiences of applying the Business Process layer to guide the design and development of SOA solutions.

The remainder of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we discuss the details of the modeling of Business Process layer. In Section 4, we introduce ABBs for this layer. In Section 5, we discuss industry experiences of applying the layer. Conclusions are drawn in Section 6.

2. Related Work

One of the most desirable features of SOA is that existing services can be seamlessly integrated to quickly compose new applications [1]. While many services are published onto the Internet on the daily basis, business process composers are facing a significant issue of how to find an appropriate service among a set of services providing the same functionalities with different features. This problem is usually known as automatic services composition [6].

A significant amount of research work has been conducted and has provided a variety of methods to tackle the problem. Some researchers use AI planning techniques [7]. For example, Zhang et al. [8] combine Hierarchical Task Network (HTN) planning approach with Partial-Order Planning (POP) and use action decomposition for plan refinements. Some researchers focus on designing algorithms of selecting optimal composition based on quality of services (QoS) features and requirements. Zeng et al. propose a middleware for making composition decisions based on four QoS factors: cost, execution duration, reliability, and reputation [9]. Liu et al. [10] propose a Web services selection algorithm using global

QoS optimization. In contrast to the individual aspects of theoretical studies of QoS-driven Web services composition, our approach offers a set of industry experience-based architectural building blocks template to guide and govern business process composition from available services.

Kim and Doh [11] propose a three-tier framework (presentation tier, service tier, and application tier) for SOA-based solution design. However, their modeling currently stays at a rudimentary level. In contrast with their high-level model, our modeling is architecture-centric model with a proved guidance for large-scale SOA solution modeling and design.

3. Business Process layer modeling

As shown in Figure 1, the Business Process layer leverages the Service layer to quickly compose and choreograph services and to coordinate business processes to fulfill customer requirements. Visual flow composition tools such as WebSphere Business Modeler (WBM) [12] can be used for design of business process modeling.

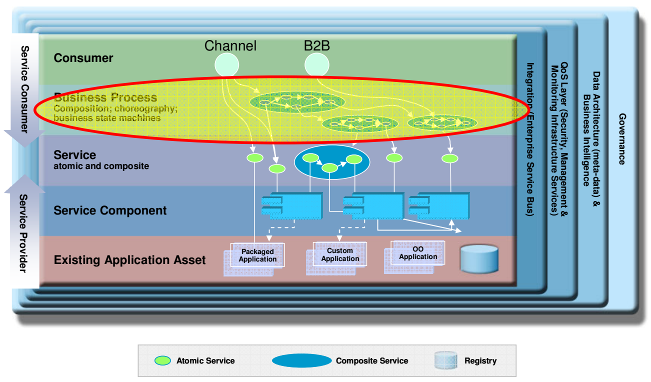
Service Consumer
Service Provider
Consumer
Channel
B2B
Business Process
Composition; choreography; business state machines
Service
atomic and composite
Service Component
Existing Application Asset
Packaged Application
Custom Application
OO Application
Integration (Enterprise Service Bus)
QoS Layer (Security, Management & Monitoring Infrastructure Services)
Data Architecture (meta-data) & Business Intelligence
Governance
Atomic Service
Composite Service
Registry

Figure 1. Business Process layer in S3.

The Business Process layer performs three-dimensional process-level handling: top-down, bottom-up, and horizontal. From the top-down direction, the layer provides facilities to decompose a business process into service clusters (i.e., conceptual services) [1] that fulfill business functions. From the bottom-up direction, the layer provides facilities to compose existing business processes, services, and service components into new business processes. From the horizontal direction, the layer provides services-oriented collaboration control between business processes, services, and service components.

3.1 Layer description

The Business Process layer handles all business logic regarding service composition and decomposition. For service composition, this layer leverages the underlying Service layer to quickly integrate and compose services and to coordinate business processes. For service decomposition, this layer provides facilities to decompose business requirements into tasks comprising conceptual service clusters, each being realized by existing business processes, services, and service components. It should be noted that the Business Process layer does not focus on individual business process representation, which can be fulfilled by workflow description languages such as Business Process Execution Language for Web Services (BPEL4WS). Rather, this layer focuses on enabling the collaborations among business processes in an SOA solution. For example, the layer may take 10 existing business processes and aggregate them into 3 big processes, while taking charge of the collaboration between them.

This layer covers the process representation, composition methods, and building blocks for aggregating loosely coupled services as a sequencing process aligned with business goals. Data flow and control flow are used to enable interactions between services and business processes. The interaction may exist within an enterprise or across multiple enterprises. This layer also includes information exchange flow between participants (individual users and business entities), resources, and processes in a variety of forms to achieve business goals. Most of the exchanged information may also include non-structured and non-transactional messages. The business logic is used to form service flows as parallel tasks or sequential tasks based on business rules, policies, and other business requirements.

From the data flow perspective, the business context and meta data are used to support the aggregation of services within an enterprise for business process orchestration or across multiple enterprises for business process choreography.

The lifecycle management for business process orchestration and choreography are also covered in this layer. In addition to the run-time process engine (e.g., BPEL4WS engine), this layer will cover all aspects of composition, collaboration, compliance, process library, process service, and invocation elements. The details will be described in the architecture building blocks section.

3.2 Value proposition

Building reusable process blocks on demand allows the separation from enabling technologies. In fact, a business process captures a set of activities needed to accomplish a certain business goal. In today's business solutions, a business process has played a central role in bridging the gap between business and IT professionals.

From the top-down approach, a business process can be defined by business people based on customers' requirements. In order to optimize the business process for better IT implementation, a business process should be componentized as reusable services that can be modeled, analyzed, and optimized based on business requirements such as QoS (e.g., historical data described in the QoS layer), flow preference, price, time of delivery, and customer preferences. From the bottom-up approach, after a set of assets are created or identified, they could be leveraged in a meaningful business context to satisfy customer requirements. The flexibility and extensibility of services composition guided by business requirements and composition rules enable business process on demand for addressing different types of customer pain points by reusing services assets.

From interaction perspective, the Business Process layer interacts with the Consumer layer (a.k.a. presentation layer) to communicate inputs and results with role players (e.g., end users, decision makers, system administrators, etc.) through Web portals or business-to-business (B2B) programs. Most of the control flow messages and data flow messages of the business process may be routed and transformed through the Integration layer. The contents of the messages could be defined by the Data Architecture layer. The Key Performance Indicators (KPIs) for each task or process could be defined in the QoS layer. The aggregation of services could be guided by the Governance layer. All the services should be represented and described by the Service layer; and service components are represented by the Service Component layer.

From a technical perspective, dynamic and automatic business process composition poses the following critical challenges to researchers and practitioners in the field of Services Computing [1, 13]. First, existing Web services-based business process description languages do not adequately accommodate detailed requirement specification, which fact makes it difficult to create optimal business process compositions. Second, present Web services specifications generally lack a facility to define comprehensive relationships among business entities, business services, and operations. These relationships may be important to optimize business process composition. Third, a typical business process generally requires multiple Web services to collaborate in order to serve business requirements. Therefore, each service component not only has to satisfy individual requirements, but also has to coexist with other services components in order to best fit the overall composed business process. In other words, the entire business process needs to be optimized prior to execution.

In summary, the Business Process layer in the SOA Solution Stack plays a central coordinating role in connecting multiple business processes through collaboration with the Integration layer, QoS layer, as well as the Data Architecture layer, the Services layer, and the Service Component layer. Addressing those challenging issues are being covered in this Business Process layer to further differentiate the proposed SOA Solution Stack with other conceptual reference models proposed by other vendors.

3.3 Key performance indicators

The Business Process layer provides visibility control points and mechanisms for role players involved in a particular business process to check the status of activities and performance in different granularities. The performance metrics for business process-based service-to-service collaborations include KPIs, status, exceptions, business events, escalation processes or actions, and so on.

For example, in a running business process, metrics and KPIs relating to business resources include three sets. First is operation efficiency metrics such as percentage of orders delivered to customers in full quantity at a specified time frame, or average credit approval cycle time. Second is KPIs for the value chain such as available technical resources for developing a new product, the percentage of orders received electronically, the percentage of customer calls taken on initial contact, and the number of order entry errors. Third is activity metrics such as the progress of a pricing dispute management conducted by multiple collaborators in the order-to-cash value chain (e.g., bank, service provider, and customer). All these performance metrics can be embedded in the business process representations with annotations in the Business Process layer.

At design time, the performance metrics for a business process should focus on specifications of service flow, customer preferences, and business rules, which can be used for services discovery and selection. For example, service flows specify the sequences of a set of activities realized through either parallel services or sequential services requested by customers. If parallel services are desired, the flow will specify parallel numbers and parallel tasks. If sequential services are desired, the flow will specify sequential numbers and sequential tasks in a business process.

Five major aspects are critical to define and specify service composition requirements: (1) service name, (2) preference, (3) business rules binding, (4) service relationship, and (5) event (for binding as well).

Service names specify a set of particular services to be used in a business process. Preferences include a set of name and value pairs, such as the preferred UDDI registry name and its location link.

Business rules specify how the various activities of a business process can be encoded in the form of reaction patterns. More specifically, business rules govern the selection of Web services for composing optimal business processes. In more detail, a business rule can be defined as a 5-tuple (business rule ID, relevant service name, priority, condition, and behavior). The element of condition predefines cost, time, benefit or service bonus, quality of service, and specific or preferred services. The element of behavior specifies under which circumstances a specific service should be selected. Service relationships describe the business relationships between service providers. For example, if service S_1 is selected together with service S_2, the combined cost is less than the sum of their costs, which affects the overall cost in a business process composition. A relationship can be further defined as a 4-tuple (relationship ID, source service, target service, XML-based definition of the relationship).

In addition, a business process may include multiple operations that conform to a certain invocation-sequencing rule. The event list defined in the performance metrics for designing a business process is used to capture the Event-Condition-Action (ECA) mapping for event-driven business process composition. A binding event associated with these operations in a Web service triggers an event action to be performed for evaluating the service selection. An event is further defined as a 4-tuple: (event name, event queue ID, event condition, event action).

Finally, the extensible structure of the representation of performance metrics enables the import of predefined XML files, such as FlowXML file, BusinessRuleXML file, PreferenceXML file, and ECA-XML for business flow, business rules, preferences, and event-action mappings. In other words, the performance metrics representation in the Business Process layer not only acts as a container of these existing XML formats, but also carries the annotations among the objects listed in different XML files.

4. Architectural Building Blocks

As shown in Figure 2**,** we identify ten fundamental ABBs in the Business Process layer based on industry practice: (1) process decomposition module, (2) service composition module, (3) process collaboration controller, (4) data flow, (5) process compliance module, (6) process repository, (7) process service adapter, (8) access control, (9) configuration rule, and (10) state management module.

4.1 Component descriptions

As shown in Figure 2, process decomposition module ABB, service composition module ABB, and process collaboration controller ABB belong to a generic control flow enablement ABB (in short, control flow ABB). Access control ABB, configuration rule ABB, and state management module ABB belong to a generic utility ABB.

A *process decomposition* building block is responsible for dividing a business process into smaller units, each matching to a service cluster [1].

A *service composition* building block aggregates services as a business process. It may go through a service or process discovery phase to reduce the number of available services or processes candidates, guided by performance metrics derived from customer requirements. Historical QoS data may also be considered to help select services. Noted that a combination of the locally optimal services may not be a globally optimal one; therefore, this building block should also contain a flow optimization facility. The final aggregation of services is represented in BPEL4WS or other flow markup languages.

A *process collaboration* building block handles the collaboration patterns between services and processes. Commonly used terms are defined in a dictionary or ontology. Business logic is used to define the sequences

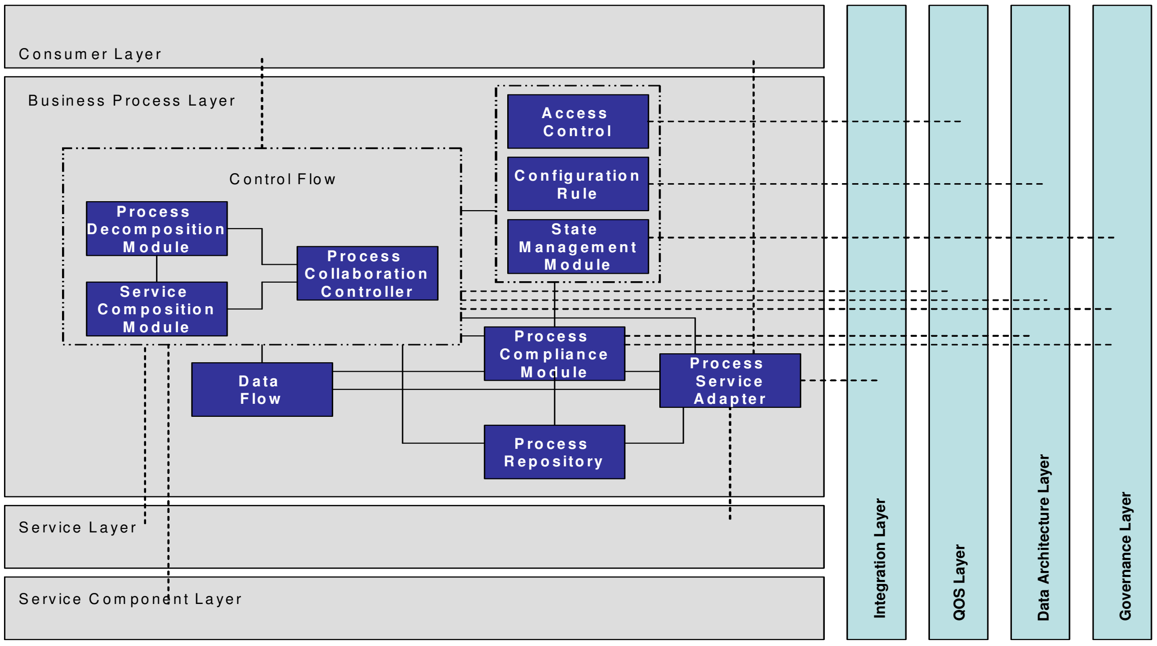

Figure 2. Business Process layer ABBs.

and contents of messages exchanged in the service-oriented collaboration environment. In more detail, message exchange patterns serve as business protocols that include predefined reusable messages, such as Request for Quote (RFQ), Request for Service (RFS), and a sequence of these messages. Regarding each message, it may include one or more message units defined by schema. All the message units are created to carry information for business resources for achieving a certain business goal.

A process collaboration controller should also contain process chorography resources, which provide the vocabulary for business collaborations. Example resources include: project, task, requirement, opportunity, IT infrastructure, and virtual team. In general, they can be represented using WS-Resources Framework (WSRF) specification. Process chorography resources can also contain cross-links to each other by endpoint references. For example, a project may refer to a set of tasks; a task may refer to a set of requirements. A set of relationships can also be defined for composition, aggregation, inheritance, and association for WS-Resources.

A *data flow* building block is responsible for managing data transactions and transformations between services and processes.

A *process compliance* building block is responsible for ensuring that a business process complies with predefined requirements or industry standards. This ABB may cover mapping, monitoring, management, law enforcement, and versioning. Mapping means that a business process is aggregated by existing services or processes. This functionality is analogical to using Partner Interface Processes (PIPs) in RosettaNet (rosettanet.org), which is an industry standards-based business process mapping. A business process should have embedded performance metrics or KPIs, so they can be monitored at run time. In addition, run-time exceptions need to be handled properly. Law enforcement means that a business process should comply with predefined rules or policies defined by local governments of agencies. For example, the annual reports of a firm must comply with the financial reporting disclosure requirements of the Sarbanes-Oxley Act (SOX) management report on internal control. In other words, all financial reporting-related business processes should be SOX-compliant. Versioning means that backward compatibility should be assured.

A *process repository* building block stores a set of business processes in a retrievable asset repository. It can be categorized by various criteria, such as by business functions or by vendors.

A *process service adapter* building block is used to externalize a business process as a service, so it can be discovered and used as a common service in the Service layer.

An *access control* building block is responsible for authorization and authentication of available business processes.

A *configuration rule* building block is responsible for hosting rules that dictate how the ABBs can be configured based on various scenarios. This unit allows the use of only appropriate ABBs. It should be noted that this configuration capability will not be effective if the ABBs are coarse grained, as it will reduce flexibility. On the other hand, if ABBs are fine grained, they will be more flexible to be configured based on specified rules. This configuration can be handled in the following two ways [3]. The first is through template-based configuration, where a user can select a specific template based on the corresponding service request scenario. The second is through dynamic template creation, where a user selects certain characteristics and the system will determine the appropriate rules and configure using relevant ABBs at

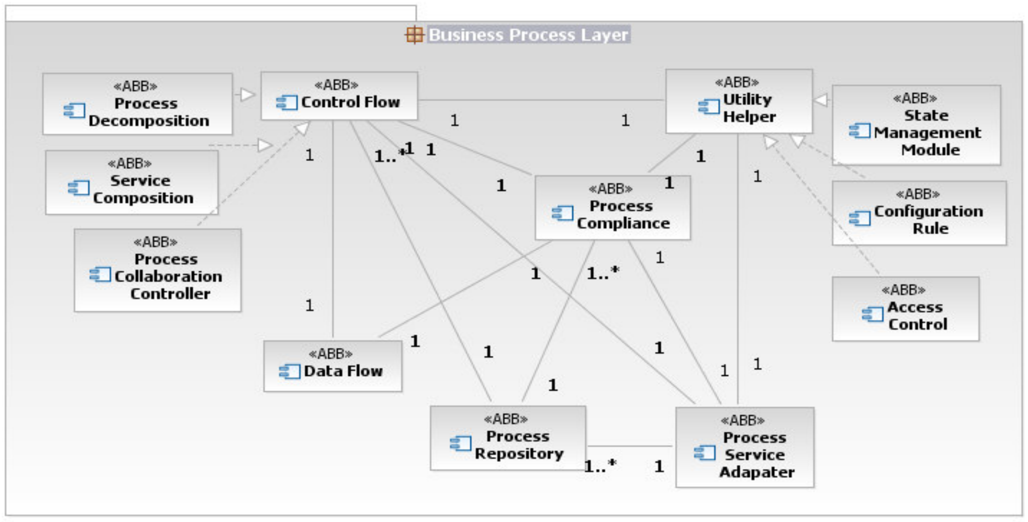

Figure 3. Component relationship diagram – ABBs within the Business Process layer.

run time.

A *state management* building block is responsible for recording and tracking business process interactions. These data typically are maintained for the duration of an entire business process.

4.2 Dependencies and interactions (patterns)

4.2.1 Relationship diagram within the layer

Figure 3 uses a UML component diagram to illustrate a static view of the relationships between the ABBs within the Business Process layer. All identified ABBs are represented as components in the diagram, with "ABB" as stereotype. The Business Process layer is represented as a package containing all identified ABBs. In order to better organize and illustrate the relationships between the ten identified ABBs, two generic ABBs are identified, as shown in Figure 3. The first is the *control flow* ABB, which is the super ABB for the three ABBs: process decomposition ABB, service composition ABB, and process collaboration ABB. The second super ABB is the Utility Helper ABB, which is the super ABB for the three ABBs: access control ABB, configuration rule ABB, and state management module ABB. With the aid of the two super ABBs, relationships between an ABB and a super ABB implies the corresponding relationships between the ABB and all the sub-ABBs. For example, as shown in Figure 3, the one-to-one relationship between process compliance ABB and control flow ABB implies three one-to-one relationships: (1) a one-to-one relationship between process compliance ABB and process decomposition ABB, (2) a one-to-one relationship between process compliance ABB and service composition ABB, and (3) a one-to-one relationship between process compliance ABB and process collaboration controller ABB.

Certain relationships exist between the ten types of ABBs: process decomposition ABB, service composition ABB, and process collaboration controller ABB realize control flow ABB. Access control ABB, configuration rule ABB, and state management module ABB realize utility helper ABB. Process compliance ABB interacts with control flow ABB, data flow ABB, utility helper ABB, process repository ABB, and process service adapter ABB. Their relationships are all one-to-one mapping relationships. In addition, there is a one-to-one relationship between control flow ABB and data flow ABB; there is a one-to-many relationship between control flow ABB and process repository ABB; there is a one-to-many relationship between process service adapter ABB and process repository ABB; there is a one-to-one mapping relationship between process service adapter ABB and utility helper ABB; and there is a one-to-one relationship between control flow ABB and utility helper ABB.

4.2.2 Relationship diagram across other layers

Figure 4 uses a UML component diagram to illustrate a static view of the relationships for the ABBs across layers. Various layers identified in the SOA-RA are represented as packages; ABBs are represented as components. Note that the relationships between ABBs within the Business Process layer are removed from Figure 4 to highlight the relationships between the ABBs across other layers.

Certain relationships exist between the ABBs in the Business Process layer with other layers. A *control flow* ABB (process decomposition ABB, service composition ABB, and process collaboration controller ABB) needs to interact with the Consumer layer (to obtain end users' inputs and other interactive information), the Service layer (to obtain services), the Service Component layer (to

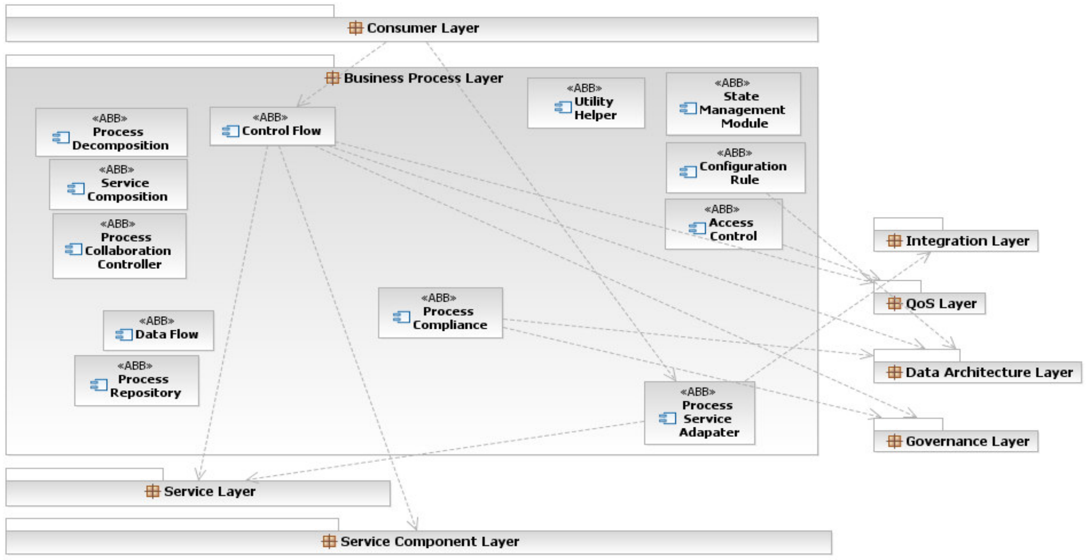

Figure 4. Component Relationship Diagram – ABBs in the Business Process layer across layers.

obtain service components), the QoS layer (to obtain QoS requirements management), the Data Architecture layer (to obtain data structure management and transformation), and the Governance layer (to obtain rules and guidance). A *process service adapter* ABB needs to interact with the Consumer layer, the Service layer, and the Integration layer. A *process compliance* ABB needs to interact with the Data Architecture layer and the Governance layer to ensure proper input/output parameter transformation. An *access control* ABB needs to interact with the QoS layer to ensure appropriate authorization/authentication control. A *configuration rule* ABB needs to interact with the Data Architecture layer to ensure proper data format management. A *state management controller* ABB needs to interact with the Governance layer to ensure proper protocol guidance.

Relationships between the ABBs and the Integration layer are extremely important, since all the message routing, transformation, and partial enablement of services composition are actually performed and realized by the Integration layer.

5. Industry Experiences

We have applied the presented Business Process layer to multiple industry projects in the last several years. In this section, we will report our findings and experiences from these project engagements.

Each project may adopt a methodology to design and manage business processes in the Business Process layer. For example, we have successful experience of leveraging the Service-Oriented Modeling and Architecture (SOMA) methodology [14] in this layer to define goal-oriented business protocols to support business process choreography. It uses service-to-service collaboration patterns and business requirements to aggregate services as a coordinated process. When a customer changes business requirements or performs business transformation using SOA, the Business Process layer leverages modeling tools to model the business processes (AS IS and TO BE) and reflects the business changes in the message exchange protocols and supporting infrastructure.

From the design decision perspective, when messages become less transactional in nature (e.g., interactions between parties change from well defined messages, to more referential requests or notifications), the interactions between Business Process layer and Integration layer may become frequent. The business process may become message- or content-driven workflow instead of a predefined sequence of tasks. From the state management perspective, the state can be managed at a global process level or at an individual service or sub-process level.

5.1 Guidance for business process composition

From the project engagements, we have summarized the following six guidance rules, aiming to enable and facilitate dynamic and automatic business process composition using existing services:

G1: A uniform representation of business requirements

It is necessary to prepare a uniform representation to precisely capture comprehensive business requirements, preferences, services features, event-action mapping, as well as the relationships among services.

G2: An automated mechanism to discover services

It is necessary to adopt an automated mechanism to generate search scripts, which can dynamically discover appropriate Web services for a specific task from UDDI registries or other Web services registries populated with Web services records in the realization phase.

G3: A seamless integration mechanism for services

It is necessary to adopt a seamless integration mechanism that can perform template-based and event-driven business process flow composition of existing services.

G4: An effective service selection mechanism for business process optimization

It is necessary to adopt an effective service selection mechanism with analytic algorithms, which can automatically construct an optimal business process using available services based on business goals and requirements.

G5: An efficient tool to support dynamic adaptation of services flow

It is necessary to adopt an efficient tool to support dynamic adaptation of services flow in regards to various modeling languages (e.g., BPEL4WS, BPMN, UML) for rapid integration.

G6: A stepwise methodology to guide business process composition

It is necessary to adopt a stepwise methodology to guide through services-based business process composition and adaptation.

5.2 Activities

The techniques and technologies in the field of business process modeling and integration can be used to model, assemble, deploy, and manage service-oriented business process in this process layer. For example, we can use IBM WBM to model the business processes. Then the activities defined in the business process maps to a set of business primitives (name and message exchange sequences) represented in a BPEL4WS format. In addition, this layer needs to create customized data entities and messages for different business scenarios based on a plug-and-play framework in this layer.

The business processes in this layer can be deployed on a process-engine-based server or embedded in screen-flow-based Web applications. In addition to the state management in the business process, runtime monitoring and tracking the status of the projects, tasks, documents or files in SOA environments are very important. It can keep a long-running process on the right track by handling business exceptions through people involvement or a dashboard-based interaction system.

6. Conclusions

In this paper, we presented our design and development of the architectural building blocks of the Business Process layer in the S3 model. Based on the industry practice, ABB modeling and templates provide a starting point for software architects to quickly configure and design a prototype for a specific SOA solution on top of the S3 model.

Although our ABB-based modeling is currently implemented using the UML technique, it is not limited to the method. Currently, the proposed componentization of the Business Process layer in the SOA Reference Architecture is part of the IBM SOMA Modeling Environment (SOMA-ME) [15], which is being used by a variety of industry projects as the core tool to conduct SOA solutioning services.

For our future study, we plan to study formalized business process collaboration techniques and algorithms, as well as business process compliance control techniques in the service oriented environments.

7. References

[1] L.-J. Zhang, J. Zhang, and H. Cai, *Services Computing*. Springer, 2007.
[2] A. Arsanjani, L.-J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, "S3: A Service-Oriented Reference Architecture", *IT Professional*, May, 2007: pp. 10-17.
[3] L.-J. Zhang, J. Zhang, and A. Allam, "A Method and Case Study of Designing Presentation Module in an SOA-based Solution Using Configurable Architectural Building Blocks (ABBs)", in Proceedings of *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC 2008)*, Jul. 8-11, 2008, Honolulu, HI, USA, pp. 459-467.
[4] L.-J. Zhang and J. Zhang, "Design of Service Component Layer in SOA Reference Architecture", in Proceedings of *33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC)*, Jul. 20-24, 2009, Seattle, WA, USA, pp. 474-497.
[5] L.-J. Zhang and J. Zhang, "Architecture-Driven Variation Analysis for Designing Cloud Applications", in Proceedings of *IEEE International Conference on Cloud Computing (CLOUD)*, Sep. 21-25, 2009, Bangalore, India.
[6] J. Harney and P. Doshi, "Selective Querying for Adapting Web Service Compositions using the Value of Changed Information", *IEEE Transactions on Services Computing (TSC)*, Jul.-Sep., 2008, 1(3): pp. 169-185.
[7] I. Paik, D. Maruyama, and M.N. Huhns, "A Framework for Intelligent Web Services: Combined HTN and CSP Approach", in Proceedings of *IEEE International Conference on Web Services (ICWS)*, Sep. 18-22, 2006, Chicago, IL, USA, pp. 959-962.
[8] J. Zhang, S. Zhang, J. Cao, and Y. Mou, "Improved HTN Planning Approach for Service Composition", in Proceedings of *the 2004 IEEE International Conference on Services Computing (SCC)*, 2004, Shanghai, China, pp. 609-612.
[9] L. Zeng, B. Benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition", *IEEE Transactions on Software Engineering*, May, 2004, 30(5): pp. 311-327.
[10] S.-L. Liu, Y.-X. Liu, F. Zhang, G.-F. Tang, and N. Jing, "A Dynamic Web Services Selection Algorithm with QoS Global Optimal in Web Services Composition", *Journal of Software*, 2007, 18: pp. 648-656.
[11] Y. Kim and K.-G. Doh, "Adaptable Web Services Modeling using Variability Analysis", in Proceedings of *Third 2008 International Conference on Convergence and Hybrid Information Technology (ICCIT)*, Nov. 11-13, 2008, Busan, South Korea, pp. 700-705.
[12] IBM, "WebSphere Business Modeler", Available from: http://www-01.ibm.com/software/integration/wbimodeler/.
[13] L.-J. Zhang and B. Li, "Requirements Driven Dynamic Services Composition for Web Services and Grid Solutions", *Journal of Grid Computing*, 2004, 2(2): pp. 121-140.
[14] A. Arsanjani, "Service-oriented Modeling and Architecture", 2004, Available from: http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/.
[15] L.-J. Zhang, N. Zhou, Y.-M. Chee, A. Jalaldeen, K. Ponnalagu, R.R. Sindhgatta, A. Arsanjani, and F. Bernardini, "SOMA-ME: A Platform for the Model-Driven Design of SOA Solutions", *IBM Systems Journal*, Jul., 2008, 47(3): pp. 397-413.