Model-driven synthesis of SOA solutions

J. K. Strosnider
P. Nandi
S. Kumaran
S. Ghosh
A. Arsanjani

The current approach to the design, maintenance, and governance of service-oriented architecture (SOA) solutions has focused primarily on flow-driven assembly and orchestration of reusable service components. The practical application of this approach in creating industry solutions has been limited, because flow-driven assembly and orchestration models are too rigid and static to accommodate complex, real-world business processes. Furthermore, the approach assumes a rich, easily configured library of reusable service components when in fact the development, maintenance, and governance of these libraries is difficult. An alternative approach pioneered by the IBM Research Division, model-driven business transformation (MDBT), uses a model-driven software synthesis technology to automatically generate production-quality business service components from high-level business process models. In this paper, we present the business entity life cycle analysis (BELA) technique for MDBT-based SOA solution realization and its integration into service-oriented modeling and architecture (SOMA), the end-to-end method from IBM for SOA application and solution development. BELA shifts the process-modeling paradigm from one that is centered on activities to one that is centered on entities. BELA teams process subject-matter experts with IT and data architects to identify and specify business entities and decompose business processes. Supporting synthesis tools then automatically generate the interacting business entity service components and their associated data stores and service interface definitions. We use a large-scale project as an example demonstrating the benefits of this innovation, which include an estimated 40 percent project cost reduction and an estimated 20 percent reduction in cycle time when compared with conventional SOA approaches.

INTRODUCTION

Service-oriented architecture (SOA) is broadly accepted in many industries as the preferred style of business architecture.[1] Vendors of application and infrastructure software, standards bodies, and academics have invested much effort in developing

©Copyright 2008 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of the paper must be obtained from the Editor. 0018-8670/08/$5.00 © 2008 IBM

methods and frameworks to build, maintain, and govern SOA solutions.[2] The majority of this effort has focused on flow-driven assembly and orchestration of reusable service components to quickly and inexpensively realize useful business functions. In particular, the Business Process Execution Language (BPEL)[3] is widely recognized as the de facto language for SOA orchestration. While there is a great deal of academic work in this area, industry practitioners often find difficulty in using BPEL to support flexible, scalable, and dynamic business process management.[4]

The industry response to this challenge has been to limit the use of SOA to the identification and definition of stateless services, with service composition and choreography left to the application components, which are designed and developed using conventional approaches.[5,6] Consequently, it becomes harder to change the IT implementation as business processes change. This negatively impacts business agility, one of the key promises of SOA. Further, reuse potential, another value proposition of SOA, suffers, because developing, maintaining, and governing libraries of broadly reusable service components is difficult. Reference 7 demonstrates that the root cause of these problems is the activity-centric behavior model used pervasively in SOA approaches today.

The IBM Research Division has pioneered a different approach to SOA solution realization called model-driven business transformation (MDBT)[8,9] that uses a radically different approach to identifying services, service components, and their behavior model. Instead of using activities as the key abstraction in modeling business processes, MDBT uses an entity-centric approach, in which a business process is modeled as the intersecting life cycles of a set of information entities.[10] The behavior of each information entity is determined by its life cycle, which is modeled using a finite state machine. The intersection of the life cycles results in communication between the entities involved. Thus, a communicating finite state machine (CFSM) model underpins the overall system behavior in the MDBT approach.

The CFSM model, along with the data model of the information entities, is used to automatically generate production-quality business service components.[11,12] Each business service component corresponds to an information entity in the business process model. The code that determines the behavior of a component is generated from the state machine model. The events that trigger state transitions are exposed as business services through service interfaces. Communication patterns between the entities are implemented as service invocations between the components. The CRUD (create, read, update, delete) services and persistent support for the component are implemented using the data model associated with the information entities.

Since the implementation of a service component is automated through code generation from high-level models, this approach significantly improves business agility, as it makes it easier to change the IT system behavior as business processes change. The reuse potential also improves, since it is easier to create, maintain, and manage broadly reusable service component models as opposed to code-based components.

Reference 13 presents a theoretical foundation for a CFSM-based approach to SOA solution realization. The authors argue that the services model is fundamentally different from other component models in that it requires only a partial definition of the behavior of a component. Further, they demonstrate that this partial definition is modeled using a state machine in which only those states and transitions that are relevant for this service definition are involved.

Though the MDBT approach has been employed successfully in a few projects in multiple industries,[14–16] integration of MDBT concepts into service-oriented modeling and architecture (SOMA),[5,6] the IBM end-to-end method for SOA application and solution development, will lead to their large-scale adoption. In this paper, we present business entity life cycle analysis (BELA), representing key techniques in MDBT-based SOA solution realization that shift the process-modeling paradigm from activity-centric to entity-centric. We also present details on the integration of BELA as a *capability pattern* to be introduced with SOMA 3.2.

Table 1 clarifies the position of the MDBT approach vis-à-vis existing static and dynamic SOA assembly frameworks (i.e., composite application frameworks) and SOA architecture frameworks. The IBM supply chain transformation team is applying the BELA technique and tools to applicable new

Table 1 Summary comparison of MDBT to SOA assembly frameworks and SOA architecture frameworks

Function	MDBT	SOA assembly frameworks and composite applications	SOA architecture frameworks
Basics	• Model service-oriented application and business rules with process modeling tool and MDHI design tool • Generate service components and UI • Develop integration components • Assemble system	Compose application from existing service components, adding business rules	• Generate application/ integration skeleton components • Develop components • Assemble system
Process modeling	Business-entity-driven process design	Conventional activity-based	Conventional activity-based
Model/view/ controller layers	Model and controller fully generated from BOM. Low-fidelity UI view is automatically generated for business validation. Custom view is specified in MDHI tool and generated.	Simple rules and logic managed in ontology engine. Primary logic is consumed in conventional services. Business rules in external business rules engine.	Application skeleton generated, followed by conventional coding.
Application integration layer	Same as conventional	Late binding, dynamic assembly	Application integration skeleton generated, followed by conventional coding
Data	Business entity data store: specified in process modeling tool, then fully generated	Same as conventional	Persistence shell generated, followed by conventional development
Model-driven implementation life cycle	Yes	Limited to policy changes that do not have impact associated services	Limited to architecture shell changes that do not impact encapsulated code
Reuse	BOM models, business entity service components, integration components, UI components and patterns	Components	Architecture framework, business components, integration components

development projects, including those related to critical parts work flows and fixed-price contractor sourcing applications (CSA-FP). We use the CSA-FP project as an illustrative example throughout this paper. Fixed-price contracting services constitute about one-half of the approximately $9 billion that the IBM Global Services division spends for contractors (i.e., technical services, administrative services, and business services). The CSA-FP project automates the procurement process, greatly reducing procurement costs.

The rest of this paper is organized as follows. The next section, "Business Entity Life Cycle Analysis (BELA)," introduces the BELA technique. The following section, "Business Entity Service Component," examines the anatomy of the synthesized business entity service components (BESCs). Subsequently, "MDBT Through the SOA Lens" describes the overall MDBT approach as an instance of an SOA reference architecture and then partitions the overall problem space into two design problems: (1) identification, specification, and synthesis of BESCs; and (2) specification, design, and synthesis of the view engine of the application. We introduce model-driven human interface design (MDHI) to address this second design problem. The section that follows, "MDBT Tooling Extensions and Tool Plug-ins," introduces the generic tooling required to support the approach and then describes its map-

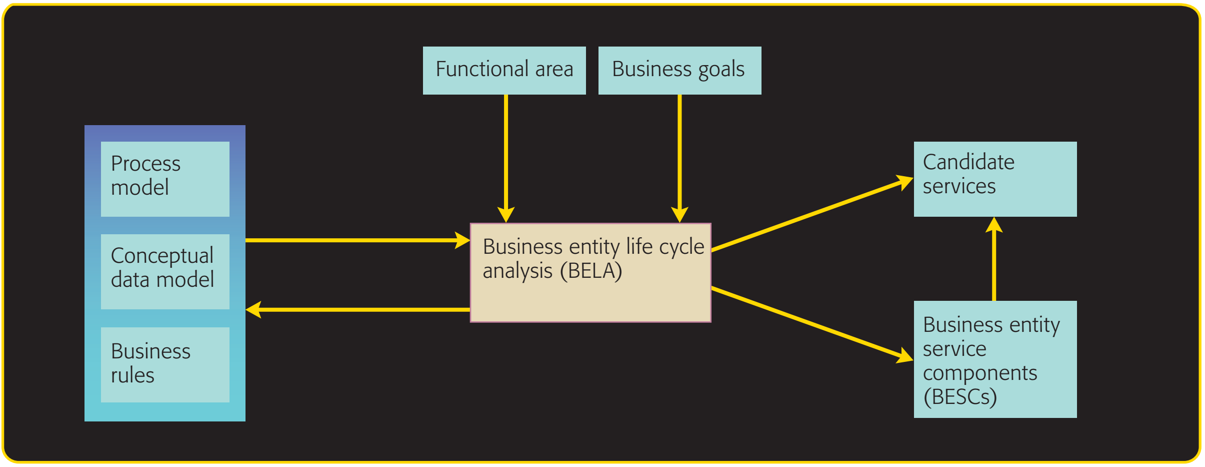


Figure 1
BELA context diagram

ping to specific IBM tools. The section titled "Summary of Benefits" summarizes the business benefits of the BELA technique (reduced cost, increased delivered functional scope, and reduced time to value). The paper ends with a summary and conclusions.

BUSINESS ENTITY LIFE CYCLE ANALYSIS (BELA)

The business entity life cycle analysis (BELA) technique uses an information-centered perspective for process modeling. Traditional, activity-centric process modeling treats information purely as a data flow construct that is input from or output to process activities and tasks. Data modelers create data models that are independent of the process models, resulting in disconnects or defects. Attempts to resolve this mismatch occur in an ad hoc manner later in the solution development life cycle. This leads to brittle solutions that fail to evolve gracefully or keep pace with changes in business requirements. In contrast, BELA uses data models actively in business process design, the identification of services, and the mapping of services to service components.

BELA will be included as a *capability pattern* with the information analysis and modeling discipline of SOMA. The high-level context diagram for BELA is shown in ***Figure 1***. In the figure, "Functional area" represents the business area in SOMA and defines the boundary in which BELA will be applied. "Business goals" and metrics are identified and decomposed in SOMA and further used to define the *scope* of process and information modeling and decomposition. Further "Business goals" provide the focus on the business entities of relevance and are an informal indicator for identifying the key business entities. "Process model," "Conceptual data model," and "Business rules" are optional inputs to BELA. The BELA technique transforms and aligns these inputs and produces updated representations leading to the identification of candidate business services and service components for further specification and realization. In their absence, BELA creates models for processes, conceptual data, and business rules.

The steps of the BELA technique are shown in ***Figure 2***. Initially, key business entities are identified. The life cycle of each entity, represented as a state machine, is derived or created (in cases where process models are not available.) The state machines are then pruned to include only elements that are business-relevant. Using the pruned state machines, process tasks and rules are mapped to state transitions. The process, rules, and information models are updated and adjusted based on the mapping. Finally, the services and component catalog is "reseeded" in accordance with the updated models.

This approach provides the necessary insight and rigor to domain analysis. Process tasks are defined with the correct level of granularity, business rules are expressed in the proper context, and logical data models are more complete and structured.

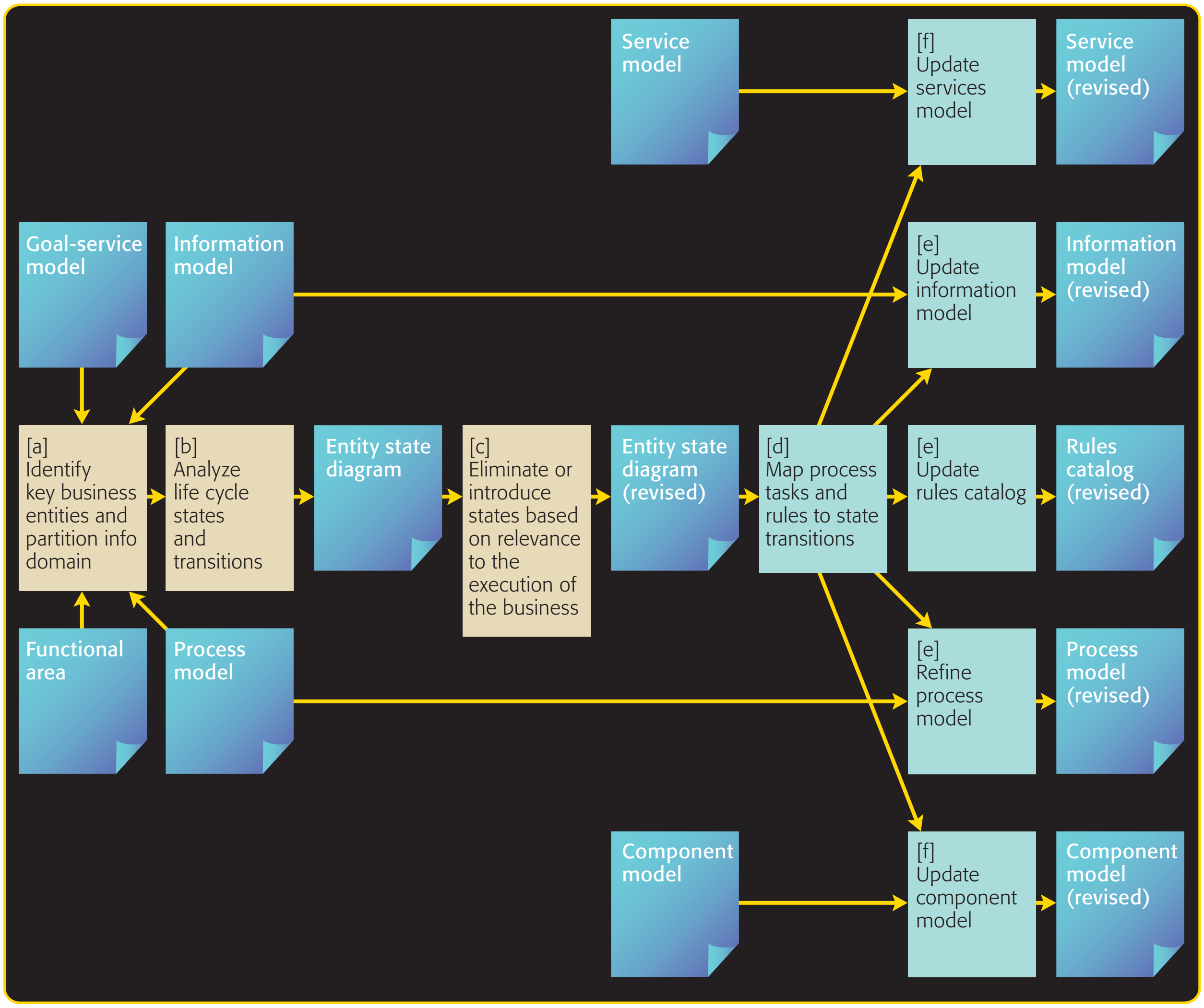

Service model
[f] Update services model
Service model (revised)
Goal-service model
Information model
[e] Update information model
Information model (revised)
[a] Identify key business entities and partition info domain
[b] Analyze life cycle states and transitions
Entity state diagram
[c] Eliminate or introduce states based on relevance to the execution of the business
Entity state diagram (revised)
[d] Map process tasks and rules to state transitions
[e] Update rules catalog
Rules catalog (revised)
Functional area
Process model
[e] Refine process model
Process model (revised)
Component model
[f] Update component model
Component model (revised)

Figure 2
BELA positioning on service-oriented method and analysis (SOMA)

The CSA-FP project started with a preexisting, conventional, activity-based process model represented with Visio** graphics software. In its first step, BELA teams the business subject-matter experts (SMEs) with IT and data architects to re-factor the existing, activity-based process model into a business entity-driven process model using a full-function process modeling tool (we used IBM WebSphere* Business Modeler [WBM]). (For cases when there is no existing activity-based process, we recommend going directly to a business entity-driven process model to reduce costs.) The team quickly identified the two business entities in CSA-FP: *fixed price request* and *supplier response*. Experience has shown that for any given application, the business entities are easily identifiable.

The team then elaborated the life cycles for each business entity. Identifying the states in the life cycles of the business entities requires careful analysis. The rule of thumb is to create new states only in cases of new increments of business value. Once the base life cycles and associated data had been modeled, business rules were added using the OMG (Object Management Group) standard Semantic of Business Vocabulary and Rules (SBVR)[17] syntax. SBVR restricts permissions based on what a given business role can do to a particular business entity.

The re-factored CSA process model, CSA BOM (business operations model), has the following advantages over the original process model. The

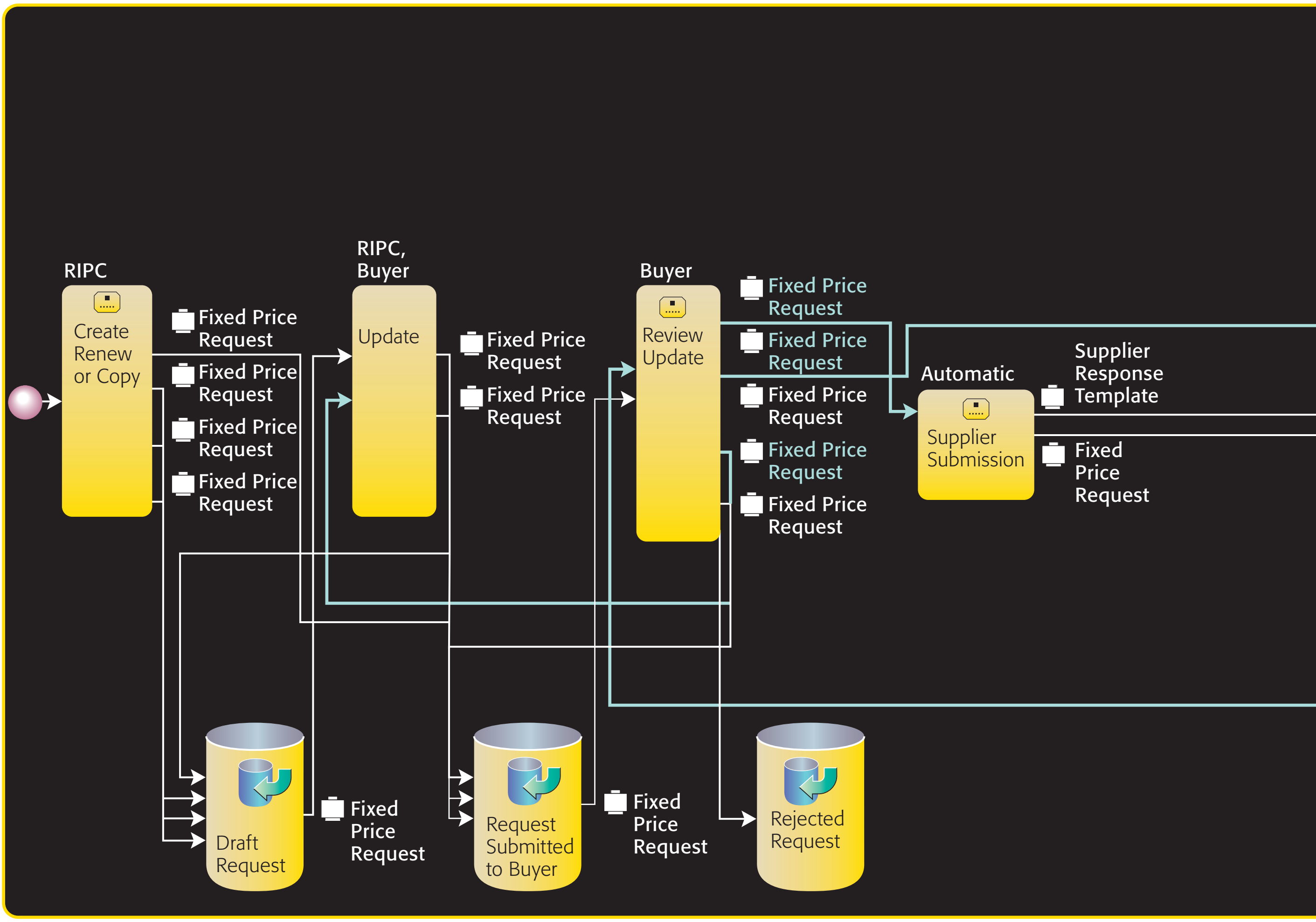


Figure 3
CSA-FP business operations model (BOM) in WBM

BOM is better structured and much easier to analyze. The BOM model filled many specification gaps exposed by BELA. The BOM process model includes a complete, explicit set of business rules. The BOM is consumable both by the SME and the IT architect. The BOM is directly usable by the IT and data architects as opposed to the original, activity-based model.

Figure 3 shows a snapshot of the BELA process model. This particular section of the model traces the fixed price request business entity from its creation (using the "create, renew, or copy" task) to the end of its life cycle up to the point when a shopping cart is created with the IBM fulfillment system, Buy on Demand (BOND). The model lays out the tasks and the roles that take the fixed price request entity through its operational milestones. The other key business entity of the process is the supplier response, as responses to the request are received from the different vendors and processed.

Table 2 provides sample rules entered using the SBVR grammar. The business rules are expressed as role-entitled conditions on the operations of the business entity (i.e., conditions that depend on the role invoking the rule). The operations are italicized.

The critical advantages of the BELA technique, as this example illustrates, include the enabling of automated synthesis of business entity service components and the partitioning of a large applica-

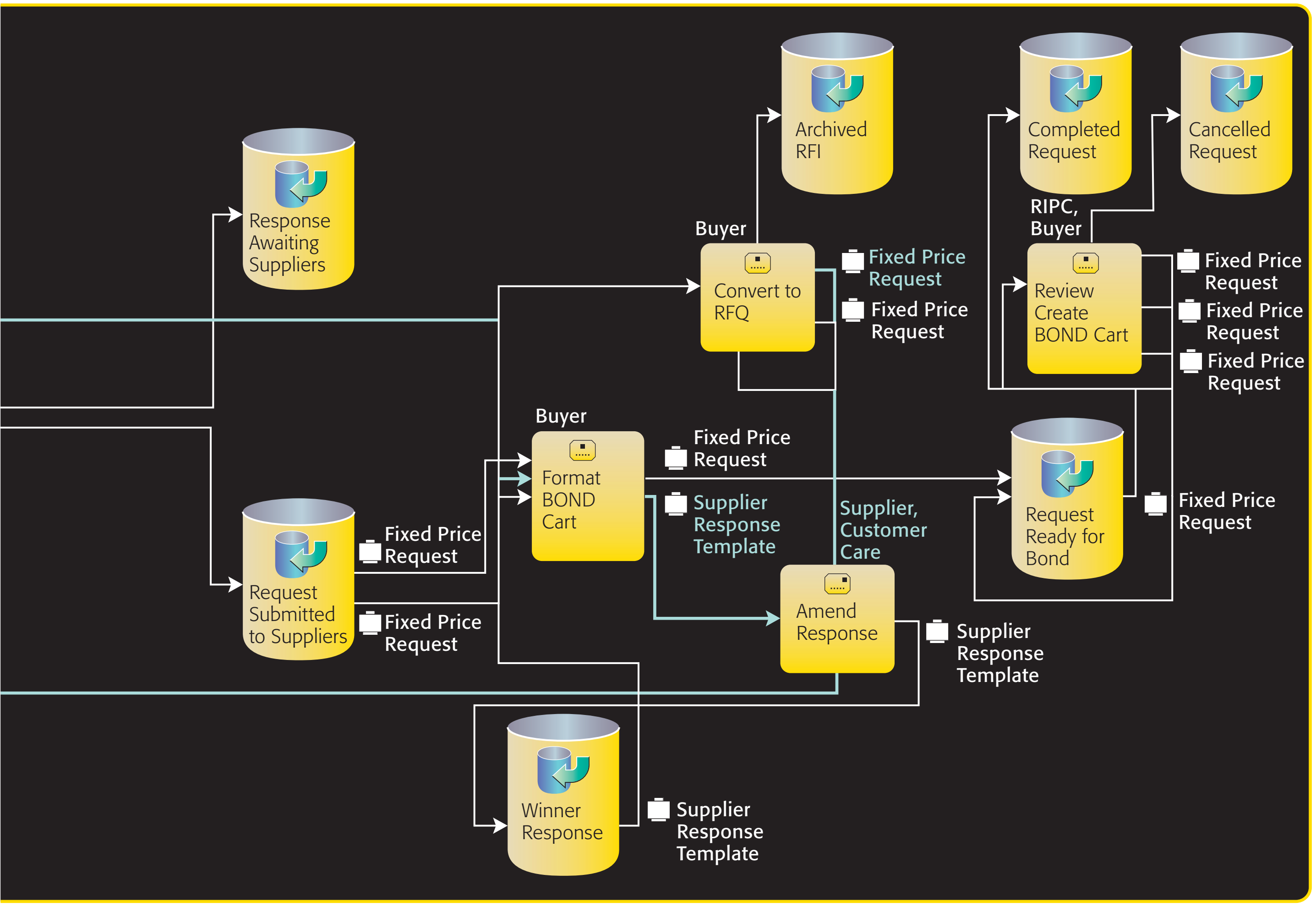


Figure 3 continued

tion UI (user interface) design problem into a set of smaller, state-transition use-case UI design problems. The business entity service components provide rich semantics as a compound composition pattern, have been field proven, and provide tremendous value toward a principled and rapid solution design. The following section examines the anatomy of business entity service components in more detail.

BUSINESS ENTITY SERVICE COMPONENT

Figure 4 illustrates the architecture of the business entity service component (BESC). The following subsections further elaborate its parts. BESC elements map to an enterprise component pattern (ECP)[18] that is leveraged to specify service components in the component specification capability pattern in SOMA.

Business element service façade

The *service façade*, as in the enterprise component pattern in SOMA, is the external interface of the entity, providing a single point of entry for business entity (BE) services to external service consumers. The BE services are categorized primarily as operational services and informational services. Operational services provide the functional view of the BEs and expose methods that trigger life cycle state changes for the entity. Informational services expose methods to create or delete instances of the BE. Additionally, queries on data owned by the BE can also be obtained through the information services.

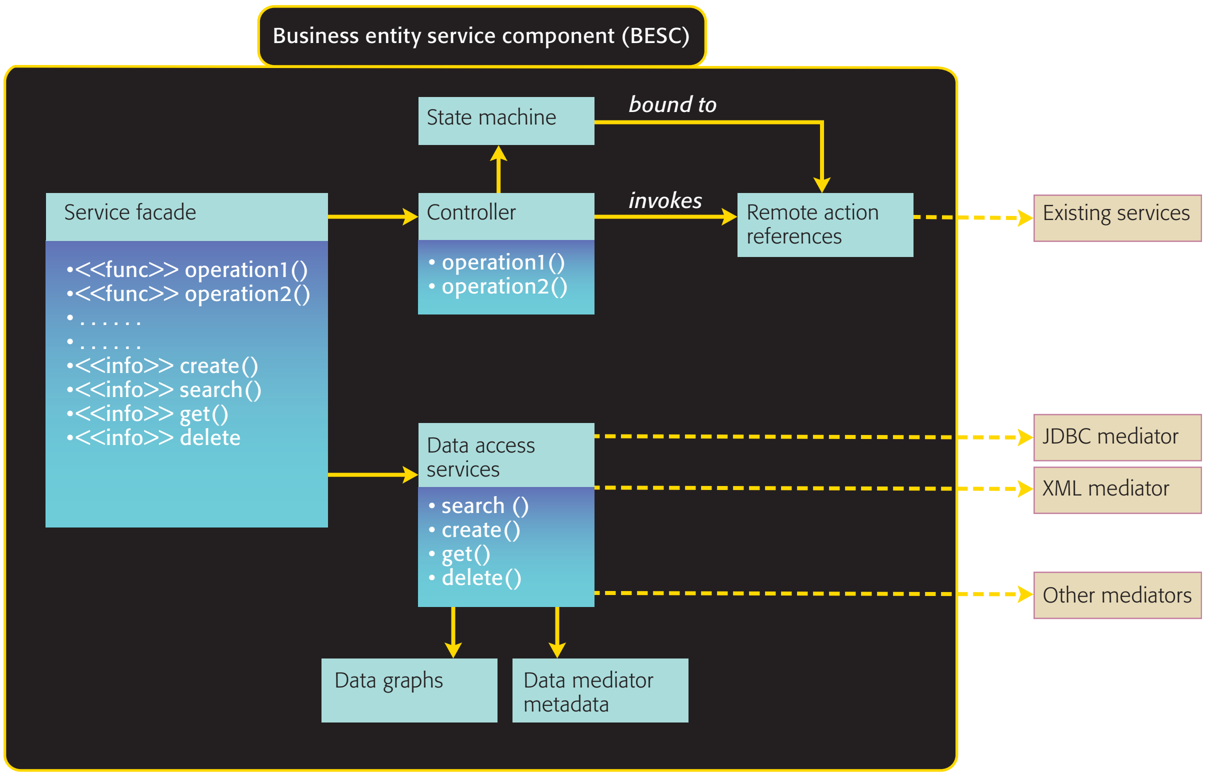


Figure 4
Business entity service component (BESC) architecture

Table 2 Sample business rules from CSA-FP expressed in SBVR grammar

Sample rules
Supplier can *withdraw or modify* fixed price request only if due date has not passed
Buyer can *return to supplier* only if fixed price request is open bid or due date has passed
Supplier can *modify* various fields of milestone only if no milestones provided by IBM or buyer permits supplier to edit them
Supplier can *submit a response* only if current date is before due date, late reply is acceptable, or buyer has changed due date
Response reviewer can *review a response* only if the supplier of the response is on the short list or is a finalist.

BE controller

The BE controller implements the behavior of the BE by reacting appropriately to invocation by the operational services. Based on the invoked operation and the current state of the entity instance, it will look up possible state transitions and execute the appropriate one after evaluating any preconditions (also known as guard conditions). It will also look up actions associated with the state transition and, if these are specified, invoke them through remote action references. On successful completion of the transition (and the completion of associated actions), the entity instance will be moved to the target state. The controller represents a mediator in the SOMA component specification.

BE state machine definition

The state machine definition represents the life cycle of the BE and is the key output of BELA. Based on this definition, the controller will react appropriately to external invocations from service consumers. The state machine definition represents a (micro) flow in the SOMA component specification.

BE remote action references

Remote action references, which represent a function mediator in SOMA component specification, are used to model actions that can be triggered while the entity is in a particular state or at a state transition. Semantically, because entity states represent process milestones, these actions denote changes to the system (or any side effects) that need to occur before the next milestone (target state) can be reached. For example, moving a purchase order BE from the "submitted" to the "approved" state could invoke an approval work flow. Only after the work flow completes (and the purchase order is approved or rejected) can the purchase order be moved to the approved (or rejected) state. In some cases, actions on a BE can trigger services invocations on another BE service component. For example, a "cancel purchase order" BESC can trigger a "cancel service invocation" service on all the associated line-item BESCs through a remote action.

BE data graphs

The data graph associated with a particular BESC is derived from the data model partitioning activity in BELA, which groups data items and arranges them hierarchically, with the root at the key business entity. This is the main data graph. However, based on application use cases to support CRUD operations, smaller data graphs are typically created as subsets of the main data graph. These smaller data graphs map to message schemas passed as input or output of informational service invocations. A common implementation of data graphs is that of the service data object (SDO).[19] The data graphs are represented as domain objects in the SOMA component specification.

BE data access services

The data access service (DAS) provides persistence support for the BESC data graphs. This service is responsible for mapping the transient and technology-generic data graphs to the selected persistent schema and invoking the appropriate data persistence mediator to update or retrieve data from persistent storage. Commonly used mediators are the JDBC** (Java** Database Connectivity) and the XML Mediator. In case of the JDBC Mediator, the DAS maps SDOs and CRUD operations to message schemas and SQL (Structured Query Language) calls, respectively. DAS represents a technical component in SOMA.

BE data mediator metadata

Metadata defines the mapping between data graphs to equivalent persistence technology. For example, for JDBC, this metadata provides the mapping of data graphs to relational schemas. The DAS reads and caches the metadata. Subsequent CRUD calls look up the appropriate map, dynamically construct the SQL calls (in case of the JDBC mediator), and execute them. Data mediator metadata represents an adaptor in the ECP.

MDBT THROUGH THE SOA LENS

The BELA technique underpins the larger MDBT approach. In this section, we examine the model-driven synthesis approach as an efficient instantiation of the SOA reference architecture.[20] We then partition the MDBT application into two composite structures using service component architecture (SCA) composites "view engine" and "model/controller." We summarize how MDBT provides model-driven synthesis for both composites.

Figure 5 overlays the key elements of MDBT over the SOA reference architecture. Essentially, the MDBT architecture is a concrete and efficient implementation of this reference architecture. The basic intent is to capture business requirements, map them efficiently onto SOA-enabled components, and deploy on SOA middleware. The uniqueness of MDBT is the BE-centric approach supported by automated synthesis of production-quality service components.

In the business domain, MDBT advocates BE-based process modeling. BELA analysis identifies services and BE service components that realize the services, and these are integrated with existing assets and components.

Figure 6 overlays the CSA-FP solution architecture over the SOA reference architecture. Business process modeling incorporates MDBT to specify the process models by analyzing and modeling the life cycle of the two key business entities, *fixed price request* and *supplier response*. Existing systems like BOND influence the analysis, as part of the BE life cycles are realized through these applications. The services layer reflects the services identified by BELA and the services identified from existing applications. DAS is used to obtain domain configuration data such as work locations, skill sets, and supplier details. The fixed price and supplier

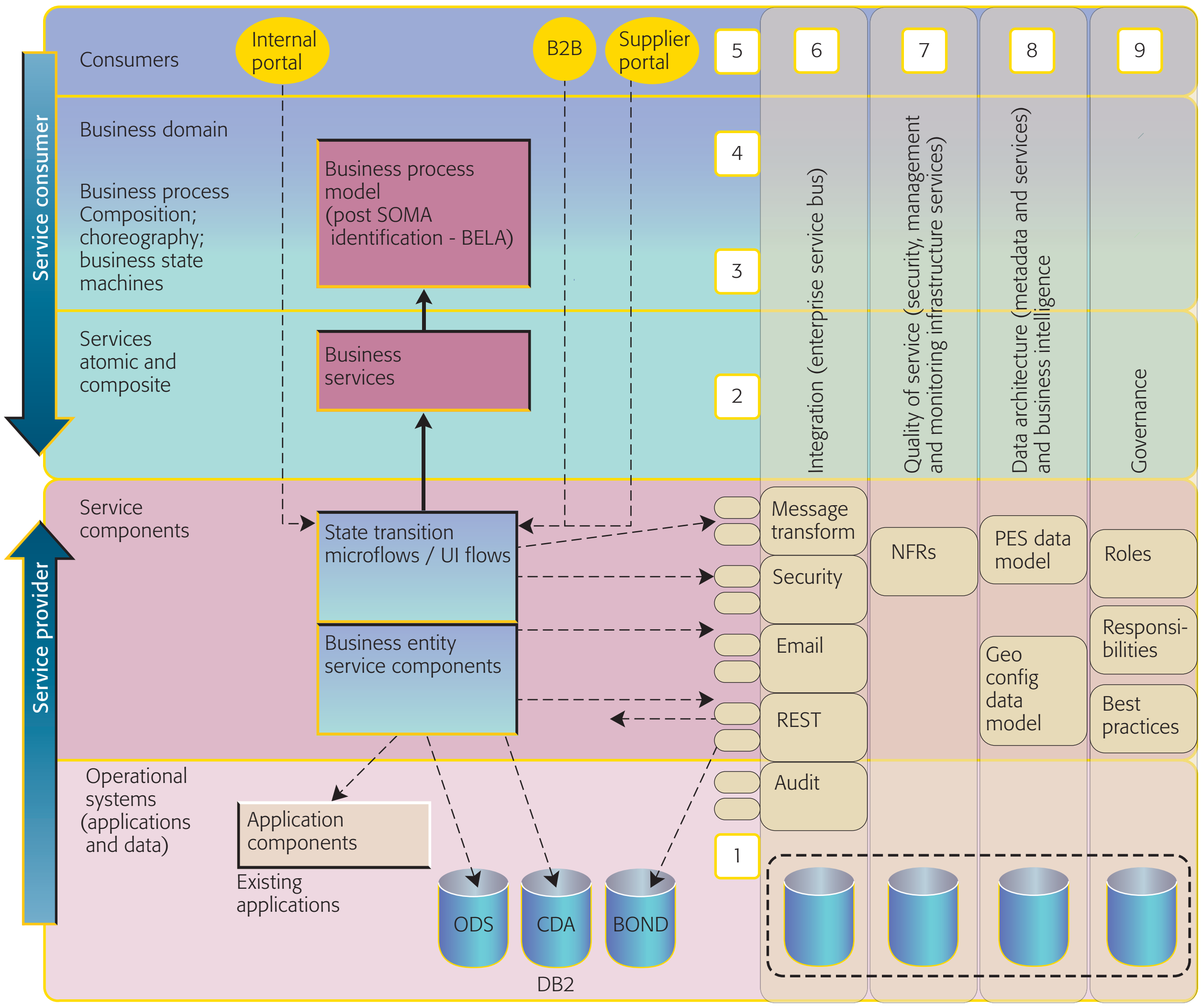

Service consumer
Service provider
Consumers
Internal portal
B2B
Supplier portal
5
6
7
8
9
Business domain
Business process Composition; choreography; business state machines
Business process model (post SOMA identification - BELA)
4
3
Services atomic and composite
Business services
2
Integration (enterprise service bus)
Quality of service (security, management and monitoring infrastructure services)
Data architecture (metadata and services) and business intelligence
Governance
Service components
State transition microflows / UI flows
Business entity service components
Message transform
Security
Email
REST
Audit
NFRs
PES data model
Geo config data model
Roles
Responsi-bilities
Best practices
Operational systems (applications and data)
Application components
Existing applications
ODS
CDA
BOND
DB2
1

Figure 5
MDBT Overlay on SOA reference architecture

response BESCs are identified from BELA and used to realize the identified services.

MDBT application architecture through the SCA lens

The SCA standard[21] provides a simple way to view SOA architectures. ***Figure 7*** illustrates the MDBT application architecture as seen through the SCA lens. We partition MDBT applications into two composites: "view engine" and "model/controller." The model/controller composite contains one or more communicating BESCs. The BELA technique, with supporting synthesis technologies, automatically generates production code for the BESCs and their associated data stores from the BOM that is fully specified in a high-level process model. The view engine composite contains the JavaServer** Pages (JSPs**) and JavaScript** that provides the UI for the application. The model-driven human interface (MDHI) provides the model-driven interface (MDI) integrating with underlying synthesis technologies for automated generation of the production-quality code.

BELA partitions a very large application UI design problem into a set of small UI design problems, one per business entity state transition. Each task in the BOM becomes a state-transition use case. The view engine composite provides the human interface for MDBT applications, encapsulating a set of BE state transition use cases. The SME works with a user-centered design professional to further partition

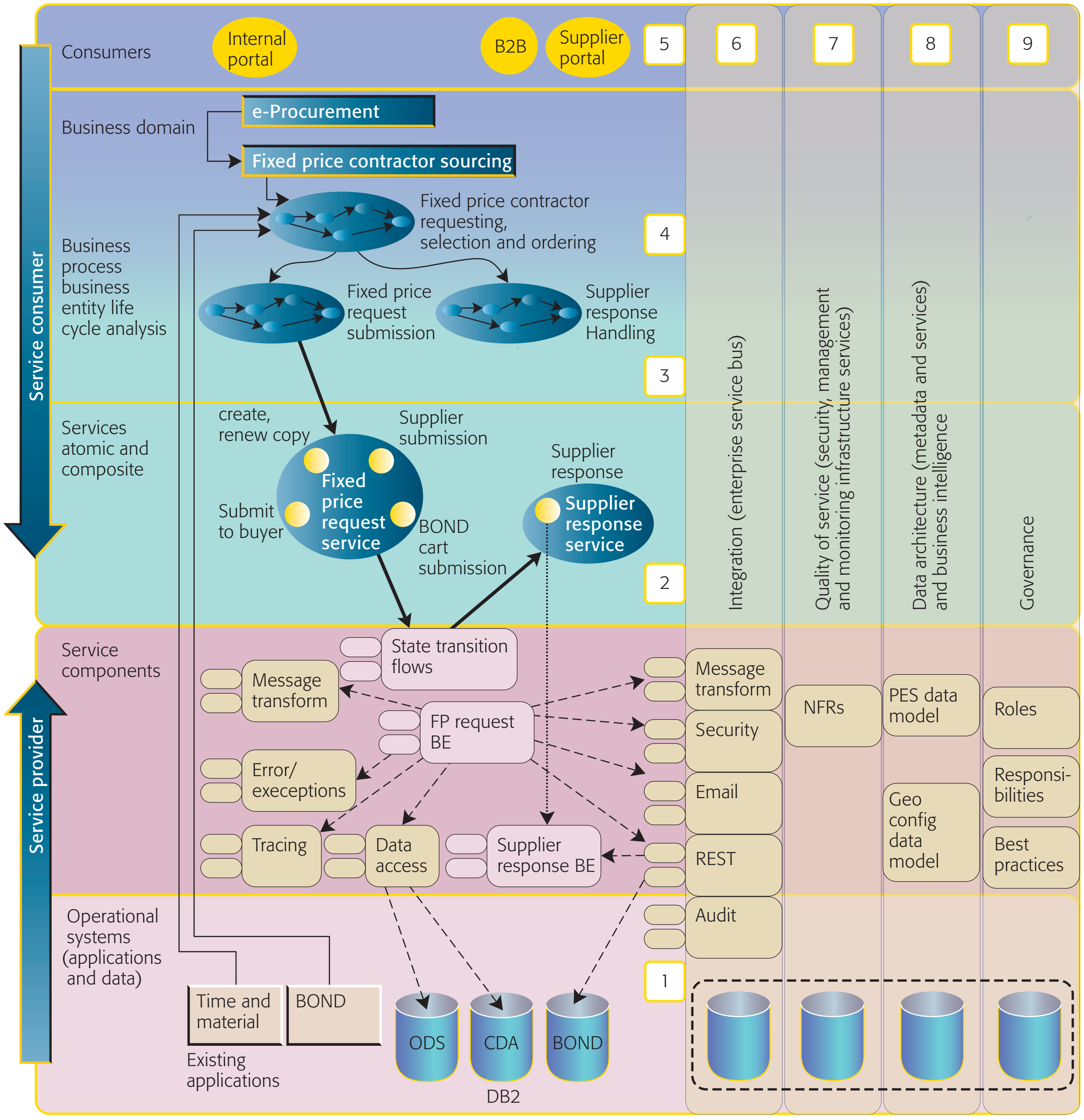


Figure 6
CSA-FP overlaid on SOA reference architecture

each state transition use case into a set of role-specific use cases using a CRUDE (create, read, update, delete, execute) matrix. The CRUDE matrix extends the traditional, data-oriented CRUD (create, read, update, delete) matrix for specifying authorization to BESC executions or functions. In Figure 7, each BE state transition use case is characterized as a component (in fact, each BE state transition use case component is realized as zero or more JSPs). The zero case corresponds to a fully automated state transition. The CSA-FP application has 20 state transition use cases. Selected state transition use cases include supplier submission, amend response, update, create new, and copy. There is an average of 12 to 15 Web pages per differentiated application role. The CSA-FP application supports three roles

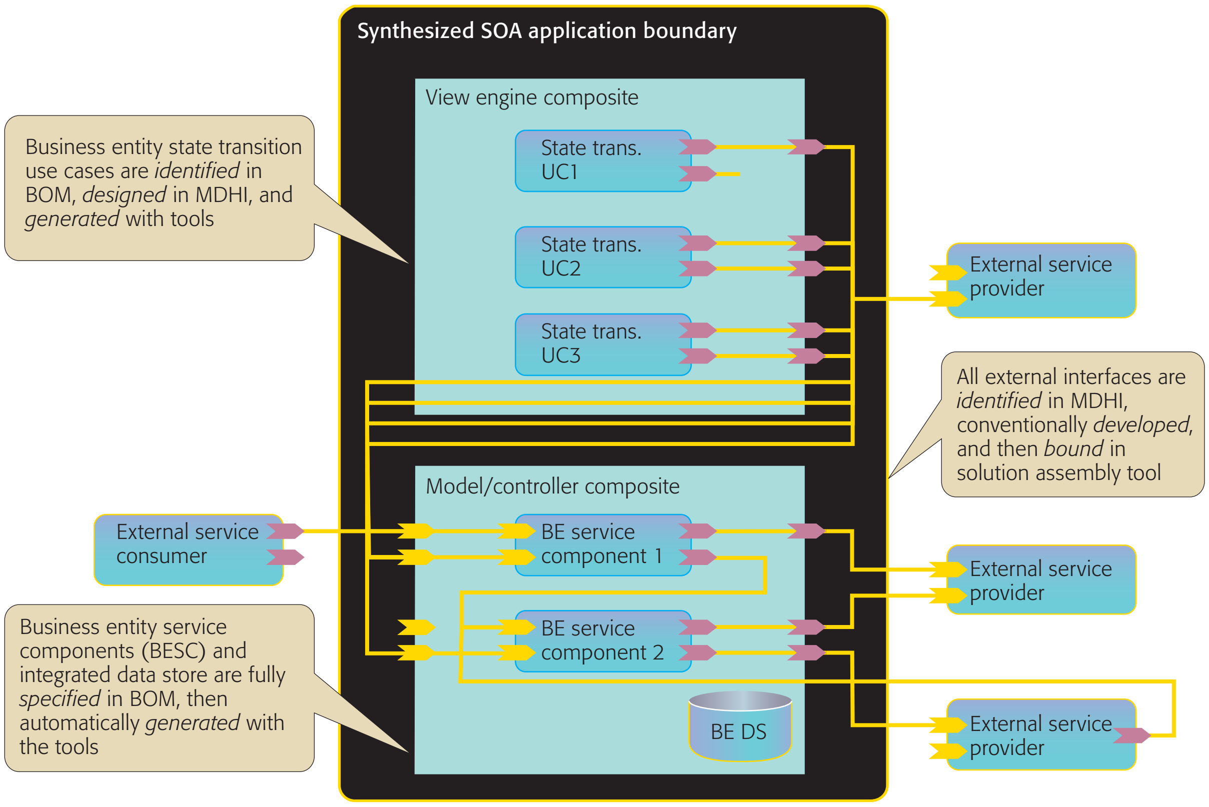

Synthesized SOA application boundary
View engine composite
State trans. UC1
State trans. UC2
State trans. UC3
Business entity state transition use cases are *identified* in BOM, *designed* in MDHI, and *generated* with tools
External service provider
All external interfaces are *identified* in MDHI, conventionally *developed*, and then *bound* in solution assembly tool
Model/controller composite
External service consumer
BE service component 1
BE service component 2
External service provider
Business entity service components (BESC) and integrated data store are fully *specified* in BOM, then automatically *generated* with the tools
BE DS
External service provider

Figure 7
Simplified MDBT application architecture through the SCA lens

corresponding to RIPC (requestor, initiator, project coordinator), buyer, and supplier.

As mentioned, BELA partitions a very large application UI design problem into a set of small UI design problems, one per state transition. Design solutions for the smaller problems are easier and more easily automated. BELA is also innovative, with the Model-Driven User Interface (MDHI) tool providing model-driven UI design with supporting code generation. While BESCs are well defined in the BOM, the page flow linkages and linkages to external interfaces are specified in MDHI tool. The MDBT tools generate service façade components for all external interfaces. The external interfaces are conventionally developed and then bound to the core MDBT application using a service assembly tool.

Actual MDBT application SCA diagrams are significantly more complicated than Figure 7. However, other than the components needed for minimal UI JavaScript, both the service components and UI components are fully model-driven, with supporting synthesis to automatically generate production-quality code. From a SOMA perspective, we identify services as part of BELA. One applies the standard SOMA service litmus test to determine which, if any, services to expose outside the application. In the following section, we examine MDBT from a tooling perspective; first from the perspective of generic SOA tools and then from that of specific IBM tools.

MDBT TOOLING EXTENSIONS AND TOOL PLUG-INS

The generic capability of the MDBT tooling stack is shown in *Figure 8*. A concrete realization of this stack with mapping to IBM software products is given at the end of this section.

Business operation modeling

Process SMEs, IT and data architects, and UCD experts collaboratively use the business operation modeling tool for BELA modeling to identify and describe the life cycles of key BEs and tasks that transition them through their life cycles. In addition, the logical data model and appropriate CRUDE role-

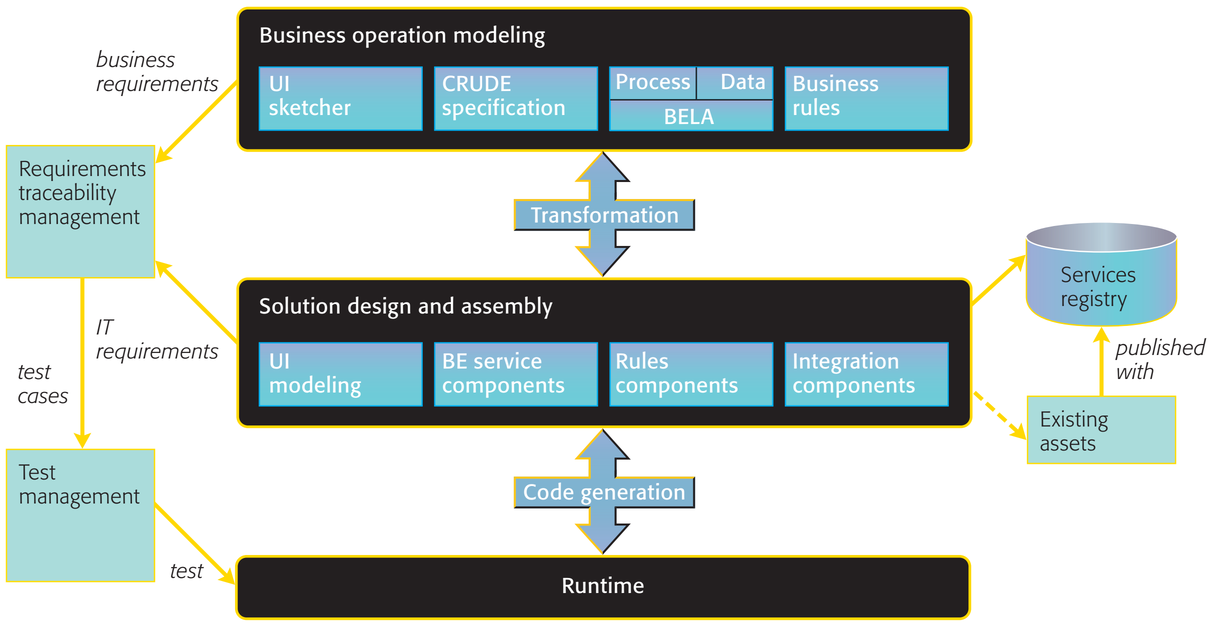


Figure 8
MDBT tooling extensions

specific constraints required to support the different tasks are specified. The resultant model is called the BOM. The UI sketching tool is used to provide the capability to create very-low-fidelity UI mockups including rudimentary layouts, data, and page flows.

The BOM model, comprising all of the above, is then transformed as per the BESC pattern presented earlier for the initial solution model. The initial solution model will have: (a) the completed BESC components with state machines and data access operations; (b) the low-fidelity UI; and (c) completed rules components. This model has enough semantics and complete syntax to generate a deployable solution without the addition of any further details. This capability to rapidly prototype is quite useful in validating the BOM.

Solution design and assembly

The solution design step extends the initial solution model with additional IT-level requirements. Typically, the extensions occur in two places: (a) low-fidelity UIs are extended toward high-fidelity UIs based on detailed UCD analysis and end-user interviews; and (b) appropriate transitions on the BESC state machines are bound to external service references either directly or indirectly through a service registry. The entire solution is then assembled with the SCA programming model and is thus made ready for deployment.

Requirements management

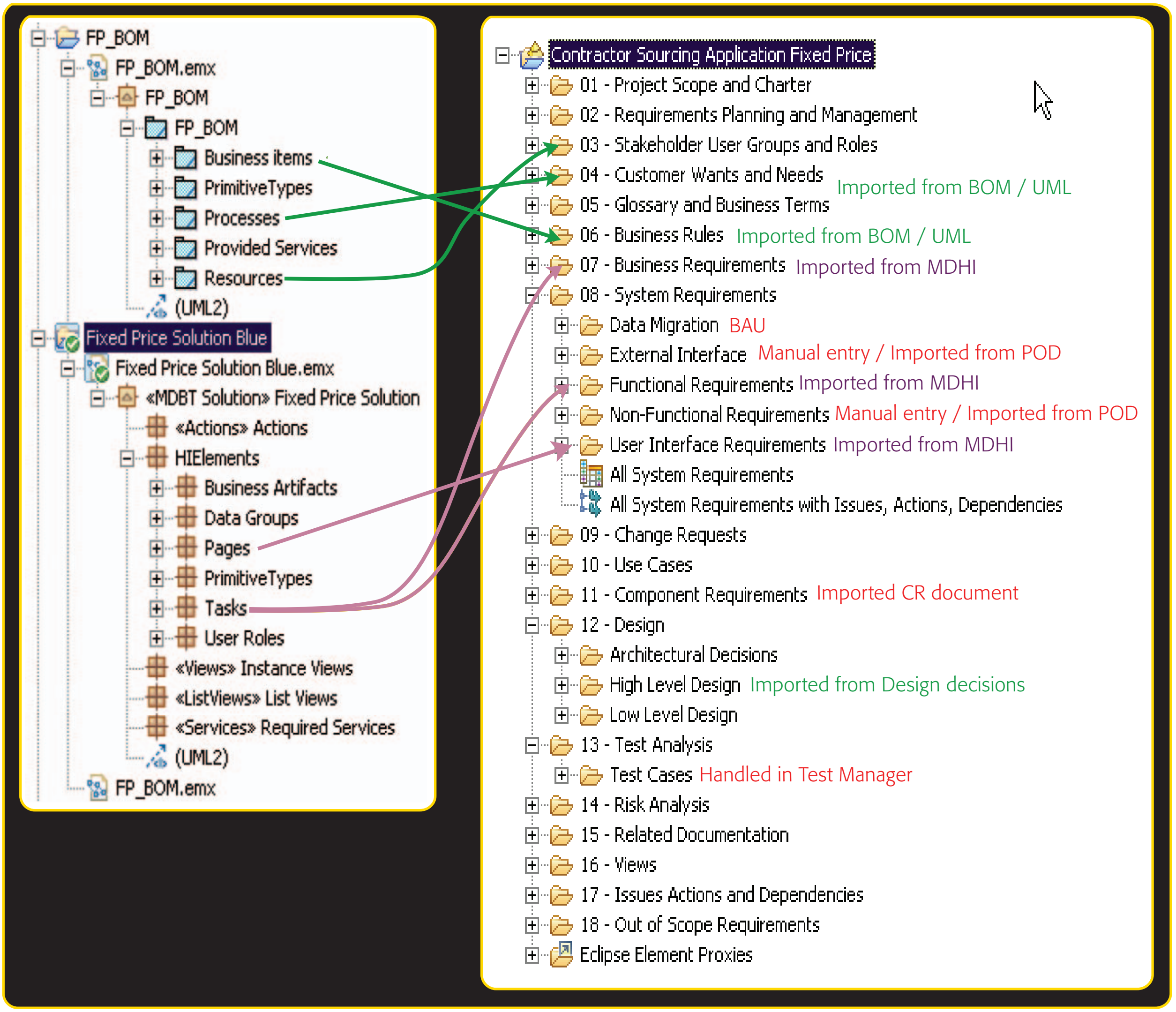
MDBT provides efficiencies in performance management. Business and solution design models bootstrap the requirements baseline (see *Figure 9*). The business processes, system use cases, business items (data), system roles, business rules, UI requirements, and component requirements are imported from business and solution design modeling tools. The external interfaces are identified but not specified.

IBM tool stack

Table 3 shows the mapping of the generic tool architecture to the IBM tool stack. Some of the advanced capabilities have been enabled with tools developed by the IBM Research Division.

SUMMARY OF BENEFITS

The value proposition of MDBT lies in four primary areas: reduced software development life cycle (SDLC) costs, increased business function scope, reduced time-to-value due to reduced cycle time from specification to realization, and reduced costs



```
FP_BOM
  FP_BOM.emx
    FP_BOM
      FP_BOM
        Business items
        PrimitiveTypes
        Processes
        Provided Services
        Resources
      (UML2)
Fixed Price Solution Blue
  Fixed Price Solution Blue.emx
    «MDBT Solution» Fixed Price Solution
      «Actions» Actions
      HIElements
        Business Artifacts
        Data Groups
        Pages
        PrimitiveTypes
        Tasks
        User Roles
      «Views» Instance Views
      «ListViews» List Views
      «Services» Required Services
      (UML2)
  FP_BOM.emx
```

```
Contractor Sourcing Application Fixed Price
  01 - Project Scope and Charter
  02 - Requirements Planning and Management
  03 - Stakeholder User Groups and Roles
  04 - Customer Wants and Needs
  05 - Glossary and Business Terms         Imported from BOM / UML
  06 - Business Rules  Imported from BOM / UML
  07 - Business Requirements  Imported from MDHI
  08 - System Requirements
    Data Migration  BAU
    External Interface  Manual entry / Imported from POD
    Functional Requirements  Imported from MDHI
    Non-Functional Requirements  Manual entry / Imported from POD
    User Interface Requirements  Imported from MDHI
    All System Requirements
    All System Requirements with Issues, Actions, Dependencies
  09 - Change Requests
  10 - Use Cases
  11 - Component Requirements  Imported CR document
  12 - Design
    Architectural Decisions
    High Level Design  Imported from Design decisions
    Low Level Design
  13 - Test Analysis
    Test Cases  Handled in Test Manager
  14 - Risk Analysis
  15 - Related Documentation
  16 - Views
  17 - Issues Actions and Dependencies
  18 - Out of Scope Requirements
  Eclipse Element Proxies
```

Figure 9
Bootstrapping requirements management

of model-based versus code-base reuse models. The following sections examine each of these areas.

Reduced SDLC costs

IBM has completed one MDBT project and has four others in various levels of completion. We use a simple SDLC differential effort/cost model to compare conventional and MDBT project costs. Project estimates are currently tracking well against this simple estimation model.

Table 4 summarizes the expected cost benefits of MDBT application development versus conventional application development. The column labeled "as is" shows the existing breakdown of effort for SDLC phases for IBM projects through the first three quarters of 2007. The column labeled "relative puts and takes" identifies differentiating MDBT cost elements within each phase. We do not expect the "model and specify" phase effort to change these cost elements significantly. The added effort to develop the more rigorous BE-driven process model is offset by more efficient requirements capture, fewer requirements gaps and less rework, and no application architecture work. The "develop/assemble" phase is where the biggest cost savings are expected, due to the elimination of core MDBT application development and unit testing.

Table 3 Mapping generic tools to IBM tool stack

Phase	Features	IBM tools
Business operation modeling	UI sketcher	Rational* Sketcher
	CRUDE support	IBM Research Division plugin to WebSphere Business Integration (WBI) Modeler
	Process modeling	WBI Modeler
	Data modeling	Rational Data Architect
	BELA support	IBM Research Division plugin to WBI Modeler
	Business Rules (SBVR)	IBM Research Division plugin to WBI Modeler
	BOM to solution design transformation	IBM Research Division plugin supporting the BESC transformation pattern
Solution Design	UI modeling	IBM Research Division plugin to Rational Software Architect
	BESCs	IBM Research Division plugin to Rational Software Architect
	Rules components	OCL (Object Constraint Language) support in Rational Software Architect
	Integration components	IBM Research plugin to Rational Software Architect
	Service registry	WebSphere Services Registry
	Code generation	IBM Research Division plug-in to Rational Software Architect
Solution assembly	SCA	WebSphere Integration Developer
Requirements traceability management		Rational Requisite Pro
Test management		Rational Functional Tester
Runtime		WebSphere Process Server, DB2*

For the "qualify" phase we expect an approximately 5 percent improvement from higher quality requirements, fewer entry defects, and quicker defect turnaround time. Conservatively, we expect about 38 percent efficiencies for MDBT projects. The CSA project cost was estimated both for conventional development and with the MDBT development, resulting in an estimate of between 30 and 40 percent expected savings. To date, the CSA project is tracking closely to this estimate. The other projects are also tracking well against this simple model.

Increased business function scope

Business SMEs for MDBT projects have been surprised at the increased functional scope this technique adds to their projects. Client satisfaction has clearly improved. It is our contention that for conventional projects, the business SMEs tend to specify their requirements in unstructured lists or activity-based process models. IT personnel put the requirements under tight change control and the business SMEs tend to disengage as the IT team interprets these requirements and creates a mapping to a solution architecture. For MDBT, the business SME is directly and intensely engaged in both the BOM development and UI design with the IT architect and UCD professional. Their requirements are directly incorporated into the models under the structured tutelage of IT, data, and UCD professionals.

Table 4 Expected MDBT system development life-cycle benefit

Phase	"As is"	Model-driven interface effort	Relative benefits and disadvantages
Modeling/specification	~24%	24%	• (+++) More rigorous BE process modeling • (-) More efficient model-based requirements capture • (——) Less costly requirements gaps and rework • (——) Application architecture predetermined • Integration architecture is as usual
Development/assembly (unit/integration testing)	~40%	8%	• (———) Development limited to integration components • Assembly is as usual • (—) No unit testing of core MDI application components • Integration testing is as usual • (—) Fewer defects
Qualification (systems integration and test)	~25%	20%	• (–) Higher quality requirements and use cases • (—) Fewer entry defects • (—) Model-based requirements gap fixed quickly
Deployment	~11%	11%	• As usual
Total	100%	62%	• Expect 38% savings for MDI projects

Note: the number of pluses or minuses shown in the relative benefits/disadvantages column indicates the relative magnitude of the benefit or disadvantage.

This process is very iterative, with successively more detail from the SMEs being incorporated into the models with each iteration. The SMEs are very confident that they are getting what they want and that it is usable. The requirements management process is fundamentally more flexible with MDBT, supporting changes much later in the cycle. As long as the changes do not affect the external interfaces or the test team, change can be supported with very lightweight change control much later in the cycle.

Reduced time-to-value

IBM is currently developing a simple SDLC cycle time model to support its strategic objective of reducing the rate and pace of change. However, at this point, we do not have an estimation model. Qualitatively, we can argue that less work translates to less time. We have established a target of reducing new development cycle time by 20 percent.

MDBT brings exciting new time-to-value opportunities to IBM for deployed MDBT applications. Specifically, for changes that do not require changes to the integration architecture of the application, development is not required. Changes to the BOM, business rules, and UI not impacting the external interfaces can very quickly be modeled and code can be generated and moved to production. We are currently classifying MDBT change classes and building processes as accelerated, model-driven test and deployment processes to quickly move new functions into production.

Reduced costs of model-based versus cost-based development

MDBT enables a new, economically attractive asset management model, incurring the cost of maintaining models versus that of maintaining code assets. By way of example, a consultant using the CSA-FP BOM and state transition use cases could engage a client, tailor the BOM and use cases to suit the client's specific needs, and generate their service-oriented CSA-FP application. Tailoring models to suit specific client needs is clearly much cheaper than tailoring code assets. Further, the cost to generalize the code assets to make them configurable across client engagements is generally prohibitive. We hypothesize that modeling assets are fundamentally cheaper and easier to maintain than code-based assets.

SUMMARY AND CONCLUSIONS

The IBM Research Division has pioneered a new approach to SOA solution realization called MDBT that uses a model-driven software synthesis technology to automatically generate production-quality business service components from high-level business process models. CFSMs provide the formal underpinnings of this technology much as CFSM formalisms underpin the success of software synthesis in the integrated circuit computer-aided design domain. Though MDBT has been employed successfully in a few projects in several industries, integration of MDBT concepts into SOMA, the IBM end-to-end method for SOA application and solution development, will lead to a larger-scale adoption. Toward that end, the BELA capability pattern, representing key techniques in MDBT-based SOA solution realization, will be integrated into SOMA 3.2.

In this paper, we have presented the BELA technique for MDBT-based SOA solution realization, a method that shifts the process modeling paradigm from one that is activity-centric to one that is entity-centric. BELA unites the process SMEs, IT personnel, and data architects to identify and specify BEs and decompose the business processes as interacting life cycles of these entities. Supporting synthesis tools then automatically generate both the interacting BE service components and their associated data stores and service interface definitions.

We have presented an MDBT project as an illustrative example and proof point for the benefits of this innovation, which include an estimated 40 percent project cost reduction and a 20 percent cycle time reduction, when compared with conventional SOA approaches.

ACKNOWLEDGMENTS

The authors would like to thank the following persons for their insight, constructive criticism, enthusiastic guidance, and help: Kerrie Holley, Ray Harishankar, and Shankar Kalyana of IBM Global Business Services; Kerin Flannery and Gautam Majumdar of the IBM Global Business Services Internal Global Account; Noi Sukaviriya, Mark Linehan, Terry Heath, Sridhar Iyengar, John Vergo, David Cohn, Senthil Mani, and Vibha Sinha of the IBM Research Division; and Jack Parsio, Blair Binney, Charles Jourdain, Mark Kemp, James Carver, Amelia Carruth, Cathy Ng-Pepe, and Russell Parks of the IBM Internal Supply Chain organization.

*Trademark, service mark, or registered trademark of International Business Machines Corporation in the United States, other countries, or both.

**Trademark, service mark, or registered trademark of Microsoft Corporation or Sun Microsystems, Inc., in the United States, other countries, or both.

CITED REFERENCES

1. D. F. Ferguson and M. L. Stockton, "Service-oriented Architecture: Programming Model and Product Architecture," *IBM Systems Journal* **44**, No. 4, 753–780 (2005).
2. *Reference Model for Service Oriented Architecture v1.0*, OASIS Standard (2006), http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html.
3. Andrews, et al. *Business Process Execution Language for Web Services, Version 1.1* (2003), http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel/ws-bpel.pdf.
4. O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg, "Service-oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned," *Proceedings of the 20th SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, ACM Press, New York (2005), pp. 301–312.
5. A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A Method for Developing Service-oriented Solutions," *IBM Systems Journal* **47**, No. 3, 377–396 (2008, this issue).
6. *Service-oriented Modeling and Architecture (SOMA)*, IBM Corporation, http://w3.tap.ibm.com/w3ki2/display/SOA/SOMA%20and%20Tools.
7. S. Kumaran, R. Liu, and F. Y. Wu, "On the Duality of Information-Centric and Activity-Centric Models of Business Processes," *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CaiSE '08)*, Lecture Notes in Computer Science **5074**, Springer, New York (June 2008), pp. 32–47.

8. S. Kumaran, “Model-Driven Enterprise,” *Proceedings of the Global Enterprise Application Integration (EAI) Summit 2004*, Banff, Canada (2004), pp. 166–180.
9. Model-driven Business Technology (MDBT), IBM Corporation, http://w3.tap.ibm.com/w3ki/display/MDBT/Home.
10. A. Nigam and N. Caswell, “Business Artifacts: An Approach to Operational Specification,” *IBM Systems Journal* **42**, No. 3, 428–445 (2003).
11. K. Bhattacharya, C. Gerede, R. Hull, R. Liu, and J. Su, “Towards Formal Analysis of Artifact-Centric Business Process Models,” *Proceedings of the 5th International Conference on Business Process Management (BPM 2007)*, Lecture Notes in Computer Science **4714**, Springer, New York (2007), pp. 288–304.
12. K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu, “Artifact-centered Operational Modeling: Lessons from Customer Engagements,” *IBM Systems Journal* **46**, No. 4, 703–721 (2007).
13. M. Broy, I. H. Krüger, and M. A. Meisinger, “A Formal Model of Services,” *ACM Transactions on Software Engineering and Methodology (TOSEM)* **16**, No. 1, 1–40 (2007).
14. R. Liu, K. Bhattacharya, and F. Y. Wu, “Modeling Business Contexture and Behavior Using Business Artifacts,” *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE ’07)*, Lecture Notes in Computer Science **4495**, Springer, New York (2007), pp. 324–339.
15. S. Kumaran, P. Nandi, T. Heath, K. Bhaskaran, and R. Das, “ADoc-oriented Programming,” *Proceedings of the 2003 Symposium on Applications and the Internet (SAINT)*, IEEE Computer Society, Los Alamitos, CA (2003), pp. 334–341.
16. P. Nandi and S. Kumaran, “Adaptive Business Objects—A New Component Model for Business Integration,” *Proceedings of the 7th International Conference on Enterprise Information Systems (ICEIS 2005)*, Miami (2005), pp. 179–188.
17. *Documents Associated with Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0*, Object Management Group (2008), http://www.omg.org/spec/SBVR/1.0/.
18. K. Levi and A. Arsanjani, “A Goal-driven Approach to Enterprise Component Identification and Specification,” *Communications of the ACM* **45**, No. 10, 45–52 (2002).
19. *Service Data Objects*, IBM Corporation, http://www.ibm.com/developerworks/library/specification/ws-sdo/.
20. A. Arsanjani, L. J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah, “S3: A Service-Oriented Reference Architecture,” *IT Professional* **9**, No. 3, 10–17 (2007).
21. G. Barber, *Service Component Architecture Home* (2007), http://www.osoa.org/display/Main/Service+Component+Architecture+Home.

Accepted for publication March 13, 2008.
Published online August 8, 2008.

Jay K. Strosnider
IBM Integrated Supply Chain (ISC), 606 Aberdeen Drive, Chapel Hill, NC 27516 (strosnid@us.ibm.com). Dr. Strosnider is the senior manager for ISC architecture. His research interest is in complex system modeling in general and business architecture modeling in particular. He has more than 50 refereed publications on a broad range of computer architecture topics from artificial intelligence to fault-tolerance.

Prabir Nandi
IBM Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598 (prabir@us.ibm.com).
Mr. Nandi, a member of the Business Informatics department at the Thomas J. Watson Research Center, received a B.E. degree in electronics and communications engineering from the Birta Institute of Technology in Ranchi, India, and an M.S. degree in computer science from the College of William and Mary. At the IBM Research Division, he has worked on business process integration and management. He coinvented the Adaptive Document (ADoc) technology and pioneered the business-artifact-centric way of modeling, composing, and implementing business process integration solutions. He is also the lead architect of the MDBT toolkit. He has authored a number of conference publications, journal articles, and patents.

Santhosh Kumaran
IBM Software Group, 294 Route 100, Somers, NY 10589 (sbk@us.ibm.com). Dr. Kumaran currently leads the banking solutions area for the IBM Software Group. Prior to this, he led the Model-Driven Business Transformation (MDBT) project in the IBM Research Division. His research interest is in using formal models to explicitly define the structure and behavior of an enterprise and employing these models to integrate, monitor, analyze, and improve its performance.

Shuvanker Ghosh
IBM Global Business Services, 3031 N. Rocky Point Drive West, Tampa, FL 33607 (sghosh@us.ibm.com). Mr. Ghosh is a member of the IBM SOA and Web Services Centers of Excellence, supporting clients in financial services and health care. He is a member of the IBM SOMA team. His interests include SOA, industry models, assets and solutions, component-based development, and distributed computing. He received a B.S. degree in mechanical engineering from the Indian Institute of Technology (IIT), India, and an M.S. degree in mechanical engineering from the University of Maryland at College Park.

Ali Arsanjani
IBM Global Business Services, 625 32nd Avenue SW, Suite 130, Cedar Rapids, IA 52404 (arsanjan@us.ibm.com).
Dr. Arsanjani is an IBM Distinguished Engineer and Chief Technical Officer for emerging SOA technologies, supporting large IBM clients in complex, distributed projects. He leads the 6000-member IBM SOA community of practice and is a member of the IEEE and the IBM Academy of Technology. ■