SOA Best Practice Report –

Architecture and Specification of Information Services with Progress DataXtend Semantic Integrator

Information services are the foundation of the successful SOA. The mature service architecture must manage an appropriate level of consistency and diversity of business information and evolve and maintain high levels of enterprise data integrity. This doesn't happen by accident. A reference architecture, policy and process framework is required together with appropriate tools and platforms that enable the essential automation and life cycle governance. In this report we describe the CBDI-SAE™ framework in context with information services and show how this can be implemented in the Progress DataXtend Semantic Integrator platform.

By David Sprott

Introduction

The application of SOA to fixing the information and data problem is one of the "next big things" for most commercial and government enterprises.

Many organizations recognize the need to reduce complexity in existing silo application portfolios, to reduce application integration effort, to establish common views of key information types such as customer, product, reference data etc. and to reduce data errors and information inconsistency.

Whilst these have been intractable issues for many years, we don't need to look far for the cause. Silo projects together with the absence of effective data management have resulted in widespread data duplication and inconsistency. See Table 1.

Typical Issues	Typical Outcomes
Responsibility for data not defined	No central data architecture responsibility - project responsibility for data No data governance; Low data quality
Inadequate SOA reference architecture	No information services policies Services combining responsibility for process and data reduce agility, increase maintenance costs Business and transformation rules spread across many layers and technologies
No canonical model	Tactical decision making; Arbitrary definition of Domains No basis for governance

Table 1 – Typical Situation Assessment

To address these fundamental issues we need to recognize two problem domains:

- Immature SOA practices, and
- Inadequate data management

Early usage of SOA concepts are usually driven by the business process. This is because early SOA activity is likely to be project specific and there is no mandate for taking a broader view. However, with experience it becomes apparent that the project and process driven perspectives are sub optimal, merely adding to existing complexity. And with that SOA experience there is likely to be emerging consensus for action to create a common services layer.

An important step towards SOA maturity is to adopt improved data management practices as part of the architecture and application delivery processes as an essential precursor to establishing common information services for core data assets. Typically these will be delivered as facades for disparate existing back end systems that publish consistent, single point of access, information services. The effective service architecture will specialize service layers, separating process from data behaviors, recognizing the inherent behavioral differences between information and process services and establishing stable information service layers that can support the more rapidly changing process layer.

For a major corporation or government department, with high complexity in the existing back end systems, the realization of the core business enterprise service layer requires a major undertaking to publish services from inconsistent, unsynchronized back end systems that span multiple architectures, delivery and runtime technologies. A very big part of this effort is managing the data inconsistencies and this usually requires a major transformation and orchestration effort together with ongoing version management that must encompass new composite applications, the core business services and the existing back end systems.

SAE Information Services Architecture Framework

The SAE[1] Reference Model and Framework (see Table 2) defines stakeholder Views in a complementary manner to other architecture frameworks such as TOGAF and Zachman. SAE is however specialized for SOA and separates out the Specification View in order to reinforce the encapsulation of the software implementation by the specified service.

Architecture View	SAE Deliverables Relevant to Information Services
Business	Concept Model
Specification	Business Type Model Specification Architecture Service Information Model Service Specification
Implementation	Service Implementation Architecture Underlying Service Specifications Automation Unit Specification (Datastore design)
Deployment	Data distribution architecture
Technology	Data Modeling tools; Meta data management tools Registry/Repository; Mediation and Transformation Services

Table 2 – SAE™ Architecture Framework – Information Services Perspective

From an information services perspective separation of the Specification and Implementation Views is essential. Information services should utilize a business oriented information model to describe service schemas and rules and ensure implementation specific data is transformed to conform to the business perspective.

In the following sections we will describe in the architecture concepts and deliverables together with supporting methodology for each of the Views and discuss how these may be supported in the Progress DXSI product set.

Introduction to Progress DataXtend Semantic Integrator

The DXSI product provides a model driven approach to data integration and service architecture. A central concept underlying DXSI is support for a common or canonical model that manages business data meanings (semantics) and facilitates standardization across disparate applications that use different data definitions. A simplified meta model of DXSI is shown in Figure 1 illustrating how the relationships between models are defined by rules based mappings.

The common model establishes the canonical data definition and provides a single source of semantics definition. The other models map to the common model. The exchange model includes the canonical, models, mappings in effect retaining the semantic relationships between source and the published data service models.

Data models or schemas can be imported from a variety of sources including UML, XML Schema and WSDL to populate source, common, data service and exchange models. Mapping between models are defined together with rules that manage data validation, aggregation, transformation and service behavior including pre-conditions.

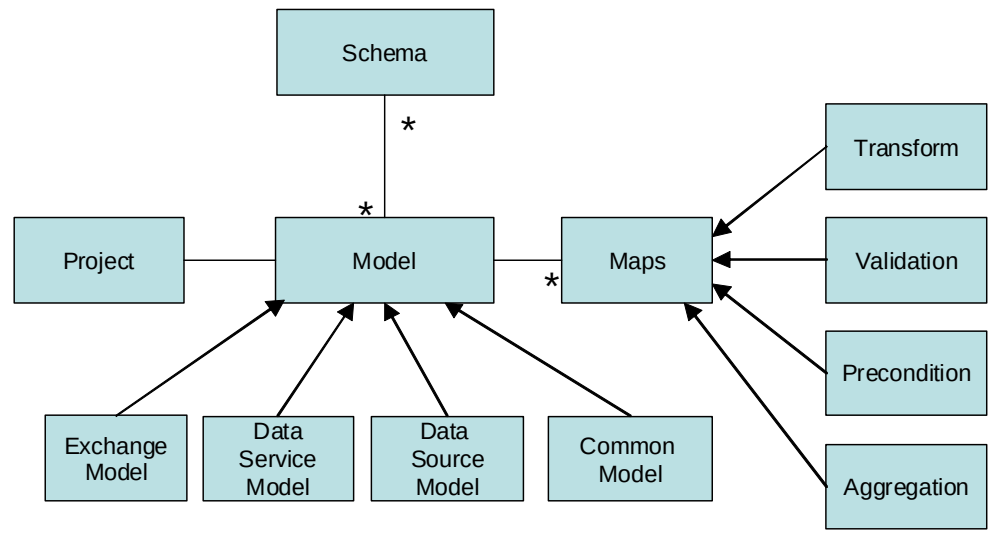

Figure 1: DXSI Outline Meta Model

From the maps and rules DXSI creates metadata and classes that implement data services. DXSI generates runtime data services that can be used anywhere a Java object can be invoked.
To implement the data services into the target environment, the SI Workbench generates wrapper code that instantiates the entities and invokes methods defined in an exchange model and Java classes. The SI Workbench generates this wrapper code for JavaBeans, SDOs and XMLBeans, which can be deployed in the application and Web servers including Apache Tomcat, BEA WebLogic, IBM WebSphere Application Server, IBM WebSphere Process Server, RedHat JBoss Application Server, Oracle Application Server and Progress Sonic.

The DXSI Workbench includes a test environment that provides import of test data and snapshots of transformations that aid debugging. Multiple versions of models can be managed.

Box 1: Overview of Progress DXSI

Business View

The SAE approach is based on understanding the business information needs of an enterprise in order to structure the foundations of the service architecture. Of course business capability and business process perspectives are also important, but in SAE they play a subordinate role in influencing the overall structure.

The question of how to capture that understanding of business information needs is of course not a trivial matter. We can all subscribe to the concept of a canonical model, but in practice there are always complexities that will need to be addressed, and we will require a methodological approach that separates levels of abstraction for defined purposes and recognize real world reality that a single common canonical model will often not be practical in a large enterprise.

Introduction to the Business Concept Model (BCM)

A distinction between levels of abstraction is common to many conventional approaches to data management. The terms Conceptual Model, Logical Model and Physical Model are often used.

However we recommend these are inappropriate for service architecture where we need to be more precise about the abstraction level that maps to the service specification, and in practice this may combine all three of the conventional model levels. More on this later. So we recommend an alternative taxonomy in order to make our purpose completely clear.

The SAE method distinguishes between the Business Concept Model (describing things as they are in the real world) and the Business Type Model (BTM, describing things as they are represented within the enterprise systems).

The BCM corresponds roughly to what might be referred to as a high-level business model, while the BTM is a business perspective model that rigorously excludes implementation detail – and therefore (at defined levels of abstraction and detail) directly maps to the service specification. Why is it important to distinguish between the BCM and the BTM? In a simple and stable situation, this distinction may just seem like an unnecessary overhead. But in many situations, we may find a significant gap between the two models, and this represents an opportunity for business improvement.

There are three popular beliefs in information modeling which are still widely accepted uncritically by many practitioners.

1. Every company in an industry should have the same data model.
2. That same data model should apply to an entire enterprise.
3. The information model remains stable over time.

Whilst it is true that information models will probably be more stable than process models, and SAE makes heavy use of this characteristic to determine the service architecture, we need to understand what information needs should be standardized and which can reasonably be differentiated within an enterprise.

When we first started teaching information modeling in the 1980s, the standard advice was to look for things-of-interest or nouns, these would be our candidate entity types. We would guide students to look for interesting-things-about-the-things-of-interest – these would be our candidate

attributes and relationships. But the problem with the traditional method of information modeling is that the notion of interest is too vague, and this often results in over simplified models that require constant extension or over generalized models which are hard to use.

So instead of a vague notion of interest, we advise using the idea of differentiation, tied specifically to events and capabilities that answer the questions – who needs to know what and when? In other words, differences in events trigger differences in response. It is this differentiated capability that allows us to decide whether business information concept should be standard or differentiated, and crucially the probability of change which should drive architecture and design decisions.

Developing the Canonical View

The term "canonical" is increasingly widely used in context with SOA to define a single authoritative or standard for data models. In fact the term is not new, my oldest reference is from James Martin [2] who said:
". . . a model of data which represents the inherent structure of that data and hence is independent of individual applications of the data and also of the software or hardware mechanisms which are employed in representing and using the data."

A high quality canonical data model is an essential capability to facilitate mature SOA. Without a canonical view, data architecture and design decisions will be delegated to individual delivery projects resulting in ad hoc agreements based upon a limited perspective. Some may argue this is preferable to the necessary time and cost investment in developing a cross organizational data model. But we advise a pragmatic approach is required; to prioritize on the business critical aspects of data standardization and, at least initially, narrowly scope the canonical model effort in order to demonstrate business value.

Consequently each enterprise will need to consider very carefully how this concept can be applied in their specific situation. Even back in 1980 James Martin advised that there may be several reasons for deviating from the canonical structure. Actually his reasons were exclusively technical, and today we know how to manage issues relating to performance and complexity. As discussed above, the issues driving deviation from the canonical structure are related to the inherent structure of the business, recognizing that certain types of data must unequivocally be canonical, but there are many data types that can be locally managed.

Better understanding of differentiation is therefore essential to developing a quality canonical model. We need to answer questions such as what aspects of the business information should be canonical? Is the canonical view required for operational consistency – for example providing a consistent customer or citizen view of their personal data. Or is consistency only required for management information purposes?

Many enterprises acquire industry models as a means of establishing a canonical model. And in some sectors there are excellent models that by virtue of wide acceptance and implementation, form a de facto industry sector standard. But whilst some enterprises use the models for cross enterprise standardization, others will use the models simply as patterns and templates to short circuit the effort in building a quality model. And perhaps most commonly in very large enterprises is a combination of both. So we may conclude that:

- It is likely that some level of “industry” standardization will apply in most enterprises, but this is highly variable.
- The industry model will always cover a subset of the data types required to support the entire enterprise. Sometimes it will be specific to core processes that are shared across other enterprises.
- There will be a mix of industry and enterprise specific standards.

Another important consideration is that information needs are not static, and it is necessary to document the differences between the AS-IS and TO-BE models in order to manage change.

DXSI and the Business View

The BCM will normally be managed in a UML or proprietary modeling environment. However the BCM is key to establishing, scoping and managing Common Models in DXSI providing governance over information management.

Recommendation: Use the BCM to define standard concepts, identify differentiated concepts and exert governance over them. The recommended BCM scope is therefore standard concepts and key differentiated concepts. Use the BCM as the mechanism to develop a narrow scope approach to data standardization, and to avoid the pitfalls inherent in delivering an ideal canonical model.

Recommendation: Maintain an AS-IS Business Concept model separate to, and as a bridge between business and IT perspectives and the primary driver for the Business Type model.

Recommendation: Formalize information governance review requirements and criteria to establish visibility over standardization and differentiation defined in the BCM.

Recommendation: Maintain TO-BE Business Concept model to support business improvement programs and manage progressive data standardization . Formalize governance review process to ensure business buy-in.

Specification View

As discussed at the outset of this report, the Specification View is an important perspective for SOA because it establishes the View that defines the published service architecture. In context to information services this is a radical departure from conventional practice in “logical data modeling” which is notoriously imprecise and largely redundant after implementation. In contrast the Specification View is based on the business level of abstraction that is used directly in service specification, and of course is an artifact that persists for the full life cycle. The question of whether this is a logical view is somewhat academic. The primary consideration is to document a comprehensive specification that allows use of the service without knowledge of the implementation. In some situations it may be necessary to communicate to the consumer what would normally be regarded as physical design decisions, perhaps in the case of performance or security related behaviors.

However we define the Specification View as:
"A logical view of the software service architecture from the Service Architect's perspective that is used to plan and specify these services from a platform independent perspective. It provides a means of thinking in depth about logical services and their interrelationships."

From an information service perspective therefore we require an information model concept that exactly matches the Service Specification and Schema. We refer to this as a Business Type Model (BTM).

Business Type Model

The BTM is a business facing information model. It can exist in varying levels of abstraction. A High Level BTM will usually be established including the main business types and most associations to support identification of business domains and other scoping activity.

A Domain BTM will be comprehensive in coverage and include most attributes, multiplicity and identifiers. This level of abstraction will support the development of the Service Specification Architecture and Service Portfolio Plan (SPP).

In addition subsets of the BTM will be realized as Service Information Models (SIM) that support the scope of individual Software Services. Any variations in details between the Domain BTM and the SIM should be restricted to clarifying any ambiguity. See Table 3.

Element	High-Level Business Type model	Domain Business Type Model	Service Information Model
Business Type	✓ the main ones	✓ all	✓ all relevant
Definition (text)	✓ yes, ideally	✓ all	✓ all
Attributes	× not needed	✓ most	✓ all
Associations	✓ most associations between included types	✓ all	✓ all
Association Name & Multiplicity	✓ useful	✓ yes	✓ yes
Identifiers	×	✓ yes	✓ yes
Attribute Optionality	×	✓ not essential	✓ yes
Attribute & Association Definition	×	× only where meaning unclear	✓ not when meaning is obvious
Derivation indicator "/" on attrs & assocs	×	✓ yes, important	✓ yes, essential
Other Invariants	×	✓ where known	✓ yes, essential
Scope ➔	Enterprise-Wide	1 Business Domain	1 Service

Table 3 –Varying Business Type Model Abstraction

How many BTMs are required? In principle there should be one logical BTM for an enterprise probably managed as a set of Domain Models. However as discussed, as a result of Business Concept modeling an enterprise may choose to standardize in certain areas and differentiate in

others. And from a practical perspective, legacy and packaged applications may also dictate that separate information models are maintained for concepts where there is low business value in alignment.

For example, a Retail Banking Division may adopt SAP Business Processes and semantics whereas a sister Retail Insurance Division may adopt semantics defined by a relevant industry model. In this case there may be a requirement for common customer concept across the enterprise, which might be restricted to simple index data that facilitates cross referral. Whereas at the divisional level it may be necessary for interoperability between disparate applications to be transformed to say our example of SAP semantics defined in the Domain BTM.

DXSI and the BTM

The BTM will usually be developed and managed as a UML model and migrated to the meta data management environment in DXSI. The BTM forms the DXSI Common Model to drive service specification and transformations.

As discussed it is likely there will be more than one BTM and hence DXSI Common Model. Table 4 explores some common situations and how varying needs for semantic consistency may be met. Note the term single refers to a logical model scope; clearly there may be multiple versions of a logical model.

Of course there will be other scenarios than those identified in Table 4. What this discourse highlights is the need to manage semantic sets as common models, and that these may be permanent (insofar as anything is permanent in an enterprise) or transitory in support of a defined roadmap that would usually be creating convergence of multiple semantic domains.

Scenario	Approach to managing semantics
1. Enterprise or major division starting to establish management and control over data, common semantics and services.	Single enterprise level Common model
2. Single enterprise, centralized data management, centralized, common business processes, common application platforms	Single enterprise level Common model
3. Single enterprise, distributed data management, silo application platforms. No business requirement to drive common business processes.	Single enterprise common model to drive common services. Common model per domain where multiple silo application platforms require semantic alignment. Single enterprise common model for management information.
4. Single enterprise, centralized data management, silo application platforms. Business requirement to drive common selected business processes.	Single common model to cover scope of selected business processes. Single enterprise common model for management information.

Table 4 - DXSI and the BTM

Scenario 1 will be the starting point for most organizations. Whilst this is an inherently enterprise initiative, it will be common for a major division to take the lead, perhaps because they have clear ownership of key enterprise concepts.

Scenario 2, where an enterprise is highly centralized, is relatively rare. But they do exist.
In Scenario 3 we will often find a drive for common core business services such as CUSTOMERS, CITIZENS, CASES, PRODUCTS, ACCOUNTS, AUTHENTICATION, IDENTITY, REFERENCE DATA and so on. But within the application silos it is equally likely there will be multiple, duplicating or overlapping applications that are contributing to various data quality related issues which can be resolved with common models that are local to the silo.

Scenario 4 highlights a further strategy that many enterprises may find useful – using a common model to align disparate information sources for management information and regulatory control purposes. It is my experience that many enterprises find almost by accident that they already have something close to a canonical model existing in support of the data warehouse. Or if this doesn't exist, there is actually a strong business case for creating a converged view for management information.

An important consideration is whether these management information common models are the same model as the operational (common) model? The answer is probably yes and no! There is a need for core operational concepts to be aligned, but also for the MI common model to define derived information in a consistent manner.

Recommendation: Formalize BTM and Common Model concepts into the enterprise reference architecture.

Recommendation: Set strategy for BTM and Common Model(s) addressing number of models, abstractions, scope and domains. Start with simple strategy based on single common model, and plan to evolve in line with progressive standardization.

Recommendation: Automate interface between UML tools and DXSI. Define versioning protocol.

Specification Architecture

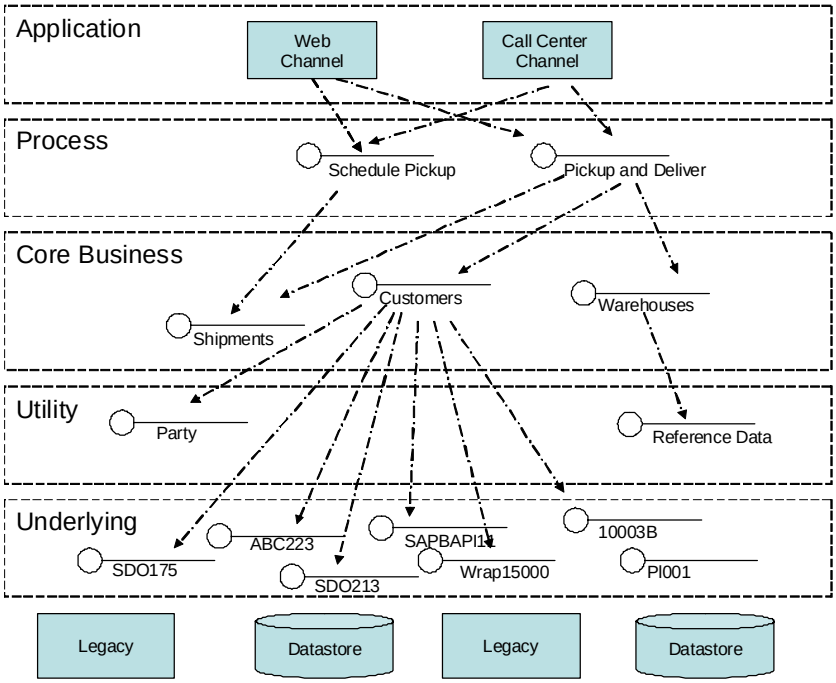

Figure 2 – Example Specification Architecture

The Specification Architecture supports the planning, specification and management of software services from a platform independent perspective. It provides a means of thinking in depth about logical services and their interrelationships. In SAE™ it is recommended that types of service behavior are separated at least at the logical level by service classifications and layering.

Experience shows this limits the impact of change in service architecture. Figure 2 shows an example loosely based on the CBDI Springfield Parcels case study which illustrates the basic reference architecture. Note there is no classification of layer named Information Service. In fact the generic term information service covers numerous behaviors and therefore in SAE™, classes of service.

SAE™ recommends these layers as a basic reference architecture. Many enterprises will customize this further, particularly by refining policies within each layer. Sub classifications may require use of specific patterns which vary depending on criteria such as performance, security, predicted agility needs and so on. Let's consider the relevance of each layer to the architecture of information services.

Architecture Layer	Behavior	Information Responsibilities
Solution Layer.	Included for completeness only.	The Solution Layer will always access information services through the service architecture.
Process Service Layer.	Orchestrates process behavior, including call to all lower service layers, plus manages process state. Generally policy will require Underlying layers to be called only indirectly, in order to only expose well formed services to the Process Layer.	Process Services will therefore not usually be responsible for managing information, except process state and transient information.
Core Business Services.	Manages Core Business Types. Implements business logic and manages the state of the business. A strong alignment between the BTM and the Core Business Services is recommended. The core business service layer may be regarded as the backbone of the service architecture, and the instantiation of the information architecture. Usually implemented initially as a façade for disparate back end applications, providing a consistent capability for common services.	Manages the state of the business
Utility Services.	The service architecture of reusable subroutines. May be algorithmic or data bearing.	Data bearing utility services may support static reference data or external services or translations such as error handling frameworks. Also may be meta data driven engines that act as dynamic information providers using meta data repository based definitions to deliver late bound information services.
Underlying Services.	Services that are difficult to consume, and which should not be called directly by the business process. Highly generic or implementation-leaking, so the interface not ideal for exposing to solution developer.	Often maintain significant data stores, and core services call underlying services to access this data & its processing rules.

Table 5 – Information Responsibility by Architecture Layer

DXSI support for Specification Architecture

DXSI supports the SAE™ classifications that provide a considerably richer architecture than the commonly deployed “data services” approach. The DXSI rules based tools support the critical requirements for high quality service specifications – the rigorous definition of the complete behavioral contract between service provider and consumer. See Table 6 below.

Class of Service	DXSI Support
Core Business Service	Rules language defines pre conditional logic Mapping to Underlying service Aggregation of multiple Underlying services Rules based orchestration of two or more Underlying Services
Utility Service	Mapping and rules defined access to external services
Underlying Service	Definition and management of Underlying Services in Source Models

Table 6 – DXSI and SAE™ Reference Architecture

Recommendation: Set policy for service specification architecture, determining patterns to be used for each class of service. Note that there may be multiple policies per service classification to cover variations such as performance, security, domain specific requirements.

Recommendation: Set policy for tool support to the specification architecture.

Service Specification

SAE™ defines a Rich Service Specification (RSS). As the name implies, this is a great deal more information than the basic WSDL. The purpose of the RSS is to provide a comprehensive specification of the behaviors provided by the service in a manner that allows a consumer to use the service without any knowledge of the underlying implementation.
If a service is to be an important, reusable enterprise asset it makes sense to create a consistent, comprehensive specification. The format of the specification may vary somewhat by class of service. For example an underlying service specification may be highly specific to the language. An inquiry service will undoubtedly be much simpler than a Core Business Service.
The RSS is a virtual concept and will be implemented in the appropriate tools, including UML tools, the Registry/Repository and DXSI. See Table 6.

RSS Section	Core Business	Underlying	Utility	Tool Support
Business Properties	✓			Registry/Repository
Technical Properties	✓	✓	✓	Registry/Repository
Quality of Service	✓	✓	✓	Registry/Repository
Standards Compliance	✓		✓	Registry/Repository
Operation Signatures	✓	✓	✓	DXSI
Mandatory Operation Sequences	✓	✓		Normally the Orchestration layer. DXSI in some circumstances. [1]

RSS Section	Core Business	Underlying	Utility	Tool Support
Service Information Model – Types and Type detail [2]	✓			UML Modeling Tool
Information Model -Invariants / Rules	✓			DXSI [3]
Operation Specifications: Properties Pre and post conditions Exceptions	✓	✓	✓	Properties: Registry/Repository Pre conditions and Exceptions : DXSI. [4]
Implementation Instructions	✓		✓	Registry/Repository
Deployment Instructions	✓		✓	Registry/Repository

Table 6 – Rich Service Specification

Table 6 Notes:

[1] It would be appropriate to manage orchestration in DXSI when mandatory sequences are based on sequencing of calls to multiple Data Sources in order to fulfill a single Data Service Operation Request (i.e. Data Orchestration), rather than true Business Process Orchestration.
[2] Service Information Model (SIM) - The SIM is a fully detailed subset of the BTM, scoped on the horizon of a service. The SIM is primarily relevant to Core Business Services.
[3] Transformation rules can be imported as XSLT.
[4] Rules language support pre-condition specification and related behavioral rules. Flexible exception handling allows standards based handling or custom, or combination of both. For DXSI there is the option of integration with sister product Sonic to handle exceptions.

Recommendation: Set policy for RSS profile and template required for each class of service and policy subset.

Recommendation: Set policy for service specification tool support by class and subset where appropriate.

Governance

Exerting governance over shared concepts is one of the harder tasks to undertake. The primary issue is well known, that shared concepts are very hard to implement because they inevitably represent some overhead for individual delivery projects.

The SAE/DXSI approach actually minimizes the overhead to a considerable extent. Specifically the SAE reference architecture and BTM approach minimizes invasive change to existing applications and the DXSI Common Model approach ensures that transforms are minimized by being mapped to the common model, not to each application/interface instance. But notwithstanding this, there is always organizational and cultural pushback because delivery dates will dominate. And program managers will call for dispensation because of mission critical deadlines, or suppliers will request dispensation because the required effort is not in their delivery contract.

The way to address this is shown in Table 7. To commence governance with business and IT management approval of the BCM, specifically focused on standardization and differentiation, linked of course to business value. All subsequent governance reviews can be traced back to this firm approval foundation.

Service Life Cycle State (Partial)	Deliverables	Type of Governance	Governance Review Criteria
Planned	AS-IS BCM	Planning	Approval of standard and differentiated data
	TO-BE BCM	Planning	Plans for evolving standard concepts
	High Level BTM	Architecture	Mapping of business to Service Specification Architecture
	Service Specification Architecture	Architecture	Approval of Core Business Services Plans for support of evolving standard concepts
		Organization	Defined ownership of common concepts (Core Business Services)
Specified	BTM, SIM Service Specification	Architecture	Compliance with profile, template and tool policy
		Usage	Use of standard Core Business Services
		Usage	Use of standard semantics
Provisioned	Automation Unit Specification	Architecture	Plans for upgrade to evolving standard concepts
Published	Registry Entry	Asset Management	Service metadata published compliant with policy requirements
		Usage	Permissions allow usage conforming with standardization policy

Table 7 – Summary of Life Cycle Governance for Data and Information Services

Recommendation: Formalize policies and integrate with existing SOA governance practices to deliver business driven data management.

Conclusions

High quality information is the foundation stone upon which enterprises deliver effective business processes to customers and manage the business. For years most organizations have failed to manage data and poor information has been the result with direct impact on customer satisfaction and cost.

As an architectural style, SOA is still in its infancy, and we may expect the concept and realization to evolve over many years. The application of SOA to fixing the information and data problem is one of the "next big things" for most commercial and government enterprises.

The reason the information and data problem represents such a major issue today is that it is genuinely hard to fix. There is real tension between "model driven methods and "just do it" methods and over time the latter has dominated because of delivery deadlines.

Today the delivery deadlines have not gone away, but there is much wider awareness of the downside of tactical projects and programs. Further structured SOA methods together with effective tools have demonstrated massive productivity savings can be made. There is an act of faith to embark on a project approach that requires significant up front planning and architecture, but managed correctly the downstream test and integration effort can be massively reduced. Customers report up to 50% savings on SOA test effort, that is sometimes as high as 50% of overall development project effort.

CBDI believes tools like DXSI represent second generation SOA life cycle automation. Such tools go way beyond early SOA automation support and genuinely allow complete traceability and governance from business to implemented capability.

References

Progress Software: Progress Software Corporation (NASDAQ: PRGS) provides application infrastructure software for the development, deployment, integration and management of business applications. We strive to increase your business' effectiveness by offering you the most effective and open set of infrastructure products possible. http://web.progress.com/index.html
Progress DataXtend - http://web.progress.com/dataxtend/index.html
Everware-CBDI: Everware-CBDI is the premier, independent source of Service Oriented Architecture and Engineering guidance and implementation. Our mission is to help organizations fundamentally change the way they respond to business needs through innovation in modular architecture and service engineering practices Research Portal: www.cbdiforum.com
Corporate: www.everware-cbdi.com

---

[1] CBDI SAE™ – To assist organizations make rapid progress in strategic SOA adoption Everware-CBDI provides CBDI Service Architecture & Engineering (CBDI-SAE™), a collection of knowledge and best practice including a reference model for SOA comprising defined concepts, reference architecture and processes that provides enterprises with a structured approach and framework for SOA based on sound architectural principles for flexibility, sharing, and consistency of Services.
The SAE™ best practices provide detailed technique guidance for delivering all the models referred to in this report.
[2] Managing the Data Base Environment, Volume I. James Martin, Savant Institute, 1980

About CBDI

CBDI Forum is the Everware-CBDI research capability and portal providing independent guidance on best practice in service oriented architecture and related disciplines.

Working with F5000 enterprises and governments the CBDI Research Team is progressively developing structured methodology and reference architecture for all aspects of SOA.

CBDI Products

A CBDI Forum subscription provides an enterprise or individual with access to a unique knowledgebase, ongoing practice research , guidance materials and resources.

The CBDI Journal, published 11 times a year provides in-depth treatment of key practice issues for all roles and disciplines involved in planning, architecting, managing and delivering business solutions.

Platinum subscription – access to all CBDI research and knowledge products including the SAE Knowledgebase

Gold subscription -access to the CBDI Journal past and present, plus other premium reports and foundational eLearning materials

Consulting Services - always highly focused, either short sharp initiatives designed to accelerate the customer's current capability, or longer term mentoring relationships. In both cases our services are provided by expert practitioners whose objectives are to effect skills transfer.

Education Services – face to face and eLearning products. Options include train the trainer and customization.

Contact CBDI

For further information on CBDI products and services contact
info@cbdiforum.com