
Serviceorienterede økosystemer og deres anvendelse i den serviceorienterede arkitektur

Analyserapport



Videnskabsministeriet
November 2005

Serviceorienterede økosystemer og
deres anvendelse i den
serviceorienterede arkitektur -
Analyserapport

Udgivet af:

Ministeriet for Videnskab,
Teknologi og Udvikling
Bredgade 43
1260 København K

Telefon: 3392 9700
Fax: 3332 3501

Publikationen udleveres gratis,
så længe lager haves ved
henvendelse til:

IT- og Telestyrelsen. danmark.dk
Telefon: 1881
sp@itst.dk
www.netboghandel.dk

Publikationen kan også hentes
på Videnskabsministeriets
Hjemmeside: <http://www.vtu.dk>
ISBN (internet):

Tryk:

Oplag:
ISBN:

>

Serviceorienterede økosystemer og deres anvendelse i den serviceorienterede arkitektur

Analyserapport

Videnskabsministeriet
november 2005

Indhold

>

Forord	v
Om rapporten	v
Baggrunden for rapporten	v
The Role of Ecosystems for SOA	1
1. Introduction	1
2. Role of ecosystems	1
Ecosystems	1
Ecosystems for SOA	3
3. Benefits and challenges with ecosystems for SOA	6
Ecosystem Benefits	6
Ecosystem Challenges	7
Service Model Challenges	8
Summary	13
4. Key Ecosystems for SOA	14
Business Service Model ecosystems	14
Infrastructure/Technology ecosystems for SOA	17
5. CBDI Recommendations	21
Appendix I: Service Ecosystem Assessment	26

Forord

Om rapporten

Denne rapport fra analysefirmaet CBDI Forum, som bortset fra dette forord er på engelsk, beskriver konceptet økosystem i forhold til serviceorienteret arkitektur.

Leverandør- og branchestøttede initiativer forventes at bestemme standarder, som vil være gældende på tværs af økosystemer, der indeholder virksomheder og deres samarbejdspartnere/leverandører. Sådanne økosystemer vil måske ikke kun bestemme den tekniske arkitektur og de standarder, som anvendes til udvikling og anvendelse af services samt den underliggende implementering. Det er også muligt, at økosystemerne vil definere den forretningsmæssige servicearkitektur og tilhørende begrebsmodeller/semantik, som styrer, hvorledes services specificeres, hvilke schemaer der anvendes til dokumenter og meddelelser, og hvilke fælles processer der anvendes inden for det givne domæne.

Rapporten diskuterer forskellige økosystemer, som kan komme til at spille en væsentlig rolle inden for serviceorienteret arkitektur. På baggrund af dette foreslås metoder til at minimere afhængighed af - og dermed "lock-in" til - et enkelt økosystem.

Det er målet, at du ved at læse denne rapport kan få svar på, eller få hjælp til overvejelse af følgende punkter:

- Hvad er et økosystem i forbindelse med serviceorienteret arkitektur, og hvilke typer af økosystemer er der?
- Hvilke leverandør- og open-source-økosystemer kan komme til at spille en væsentlig rolle inden for serviceorienteret arkitektur?
- Hvordan undgås "lock-in" til et givet økosystem?

Baggrunden for rapporten

*Hvidbog om IT-arkitektur*¹, som udstikker en lang række retningslinier for digital forvaltning, peger på, at offentlige it-systemer i fremtiden skal indrettes, så de overholder principperne for en serviceorienteret arkitektur (SOA).

Efterhånden som sektorområder og enkelte myndigheder lægger strategier og implementeringsplaner for, hvorledes man bevæger sig frem mod en serviceorienteret arkitektur, opstår der en række spørgsmål, som Hvidbogen ikke nødvendigvis giver svar på. Videnskabsministeriet er derfor i gang med at lave en samlet ramme for forskellige aktiviteter på dele af arkitekturområdet, som også er relevante for serviceorienteret arkitektur. I forbindelse med etableringen af denne ramme vil indsatsen på en række områder blive skærpet med henblik på at kunne give mere konkrete anbefalinger.

Eet initiativ har været at få udarbejdet analyserapporter, som kan understøtte det fællesoffentlige it-arkitekturarbejde ved at vurdere forskellige aspekter i forbindelse med indførelsen af en serviceorienteret arkitektur.

¹ Hvidbog om IT-arkitektur kan hentes på <http://www.oio.dk/arkitektur/publikationer/hvidbog>

Det drejer sig om følgende rapporter på engelsk:

- E-Business Standards. The Role of ebXML and Web Service Protocols
- The Role of Ecosystems for SOA
- Infrastructure Services for SOA within the Public Sector
- SOA Reference Models

Videnskabsministeriet vurderer, at en stor del af indholdet i rapporterne også er af interesse for individuelle organisationer, og ministeriet offentliggør dem derfor på OIO.dk.

Det er vigtigt at notere sig, at rapporterne er udarbejdet i oktober/november 2005, og at nogle af rapporterne, eller dele af dem, relativt hurtigt kan blive utidssvarende, hvis de beskæftiger sig med områder, der er i stærk udvikling.

Analysefirmaet CBDI Forum er ene og alene ansvarlig for vurderingerne og anbefalingerne i rapporterne, men Videnskabsministeriet tilslutter sig generelt rapporternes indhold.

CBDI Report

The Role of Ecosystems for SOA

Abstract: This paper discusses the concept of an ecosystem led approach for Service Oriented Architecture (SOA). Vendor- or industry-led initiatives can be expected to set SOA standards that apply across an ecosystem of user organizations and partner vendors. Such ecosystems may establish not only the technical architecture and standards that apply to the delivery and use of Services as well as their implementation, but also the Business Service architecture and semantics that govern the Service specifications, document/message schemas and common processes for their domain.

This paper considers the various ecosystems that may emerge and suggests approaches that minimize dependencies on ecosystems to avoid “lock in”.

Contents	Page
Role of Ecosystems	1
Benefits and challenges with ecosystems for SOA	6
Key Ecosystems for SOA	14
CBDI Recommendations	21
Appendix I: Service Ecosystem Assessment	26

Version: 1.0

Date: October 2005

Author: Lawrence Wilkes.

lawrence.wilkes@cbdiforum.com



Independent Insight for Software Oriented Practice

CBDI Forum Report: The Role of Ecosystems for SOA

1. Introduction

This paper discusses the concept of an ecosystem led approach for Service Oriented Architecture (SOA). Vendor- or industry-led initiatives can be expected to set SOA standards that apply across an ecosystem of user organizations and partner vendors. Such ecosystems may establish not only the technical architecture and standards that apply to the delivery and use of Services as well as their implementation, but also the Business Service architecture and semantics that govern the Service specifications, document/message schemas and common processes for their domain.

This paper considers the various ecosystems that may emerge and suggests approaches that minimize dependencies on ecosystems to avoid “lock in”.

2. Role of ecosystems

Ecosystems

The concept of ecosystem is commonly associated with biology. For example, “... an ecosystem is a naturally occurring assemblage of organisms... living together with their environment..., functioning as a loose unit.”¹

Core to the concept of an ecosystem is that participants benefit from each other via symbiotic relationships. Participants provide benefits to each other to achieve an objective that otherwise would be difficult to achieve.

However, whilst participants in the ecosystem might benefit from the symbiotic relationship it is sometimes the case that individual participants are unable to survive outside the ecosystem, which therefore forms a constraint on their ability to adapt. Equally, the failure of one participant might lead to a failure of the whole ecosystem.

A business or Information Technology (IT) ecosystem can be seen as one where the individual participants each provide different but related products and services in a symbiotic relationship in order to further the overall market for those products and services.

In the business and IT markets, some participants of the ecosystem (or more often a single dominant participant) might use the ecosystem approach to lock-in other participants by creating dependencies on their products, service or technology. Whilst the ecosystem is growing or remains sustainable participants may benefit through having an advantage in the marketplace. As the ecosystem gets larger, it becomes difficult for others to emulate or compete.

The downside is that the costs of switching ecosystems may become prohibitive. In some markets a dominant ecosystem may not prevail, forcing many organizations to participate in multiple ecosystems which may now be seen as a cost rather than a benefit. In such an environment, lock-in to a single ecosystem can result in lost opportunity of not being able to participate in some parts of the marketplace.

The Role of Ecosystems for SOA

Type	Basis	Leadership	Participants	Example
Platform	Adoption of platform as foundation for applications	Vendor	Usage Extension	Microsoft Windows
	Platform could be OS, database, middleware, or combination of	Developer Community	Development of platform Extension	OSS Linux
Developer	Adoption of tool or language	Vendor	Usage Extension	Microsoft Visual Studio
		Developer Community	Development of language Extension	Java
Application	Adoption of application as foundation for extension	Vendor	Extension	SAP Microsoft Office
Solution Framework	Adoption of a framework comprising a mixture of platform and application components tailored towards a specific solution or industry	Vendor	Usage Extension	
Industry or Business Domain (vertical or horizontal)	Adoption of common standards and/or technology	Dominant User	Usage	Automotive Government/Public Sector
		Industry consortium	Development of standard Usage	FIX
	Use of application	Vendor	Usage	Sabre Travel Network

Table 1 - IT Ecosystem Types

Whilst a successful ecosystem may not be displaced by a competing ecosystem in the same domain, the rapid evolution of information technology may introduce new ecosystems that are formed to support a new paradigm (such as the development of a new technology) and which displace existing ecosystems by rendering them irrelevant to the new requirement. Though large and successful, the existing ecosystem(s) is unable to respond because the participants, which may include the dominant participant itself, are locked into the existing paradigm. Consequently the ecosystem and some of its participants declines or even fails.

Table 1 lists the basis for ecosystems that are normal in the IT market. Vertical Industry ecosystems might be seen as more business oriented, but ultimately still require an underlying IT-based ecosystem

Often ecosystems will exhibit a combination of these types. For example, an application or platform may be the basis of an ecosystem only within a specific vertical business – such as the Sabre Travel Network and IBM TPF in the air travel industry. Or there may be

dependencies, such as the Microsoft Visual Studio developer ecosystem is dependent on, and effectively a subset of the Microsoft Windows platform ecosystem.

Ecosystems for SOA

Rather than depending on a requirement that endpoints support a common implementation or underlying technology, SOA is based on interoperability between endpoints by agreement of a Service Model that specifies the,

- specific instances of the Services in use and their behavior
- protocols and semantics used to describe the Services offered, and the messages exchanged (payloads)
- policies that govern use or provision of the Services
- contracts that detail the obligations on the endpoints

The Service Model should be defined independently of the underlying implementation. It provides access to the capabilities provided by the implementation. Whilst a run-time the Services will be coupled to a specific implementation instance, the specification and design of the Service Model should not be constrained to a specific implementation instance.

This principle, usually referred to as “loose coupling”, promises greater agility as endpoints are not bound together by the implementation or technology. As long as they continue to abide by the Service Model, the implementation or technology that underpins each endpoint can independently change without impacting the other. Furthermore, endpoints may easily enter in to new exchanges with different endpoints or switch relationships as long as the other endpoints conform to the same Service Model.

This principle should hold true whether the endpoints in the exchange are all internal to an organization, or across a federation of two or more organizations.

The Service Model can be refined into either a

- **Business Service Model (BSM)** –a model of the Business Services for a specific business domain, such as Human Resources
- **Infrastructure Service Model (ISM)** –a model of the Infrastructure Services for a specific technology domain, such as registry, message delivery, security or manageability

The BSM connects or binds the consuming application to the providing application via the Business Services.

The ISM connects or binds the Service Consumer's and Provider's applications to their respective operational infrastructure, and additionally can bind the Service Consumer and Provider through dependencies on common infrastructure protocols and Services. For example, they must use the same Registry Service to publish and discover each others Business Services.

These concepts are illustrated in Figure 1.

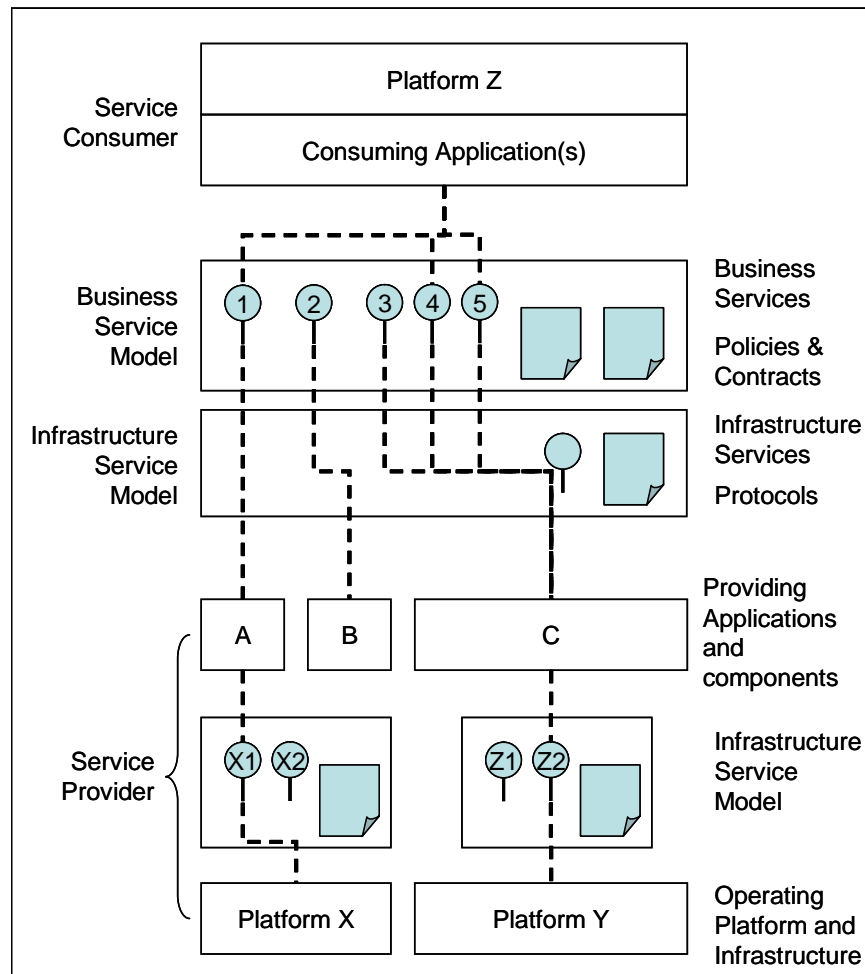


Figure 1 - Service Model Layers

IT Ecosystems for SOA can therefore be seen as relevant on different levels

SOA Service Model Ecosystems - IT ecosystems that specify a Service Model. These could be either

- **Business Service Model Ecosystem** – For example, vertical industry initiatives that specify a BSM for B2B exchanges in a particular industry such as ACORD for insurance, or a vendor initiative to provide a BSM for their packaged application such as a SAP ERP. Large ecosystems of users and vendors can grow around either
- **Infrastructure Service Model Ecosystem** – For example, industry initiatives that specify protocols and infrastructure Services required to support B2B interoperability such as ebXML or WS-STAR (Web Service Protocols for secure, reliable transactions), or for platform virtualization such as grid OGSA. Or initiatives that specify an ISM to provide access to a specific infrastructure resource, such as security, manageability, or persistence.

SOA Service Model Implementation Ecosystems - IT ecosystems that provide an implementation of the Service Model, whether BSM or ISM. This is normally a vendor led ecosystem, though open source implementations may also be available.

Which takes precedent, the model or the implementation, will vary for a number of reasons. In an industry initiatives such as ebXML, the model comes first. That is, the implementation

The Role of Ecosystems for SOA

is designed to support the model, and vendors or users build their implementation based on the model. Whereas with some vendor initiatives, the implementation comes first. That is, the model is designed to support the implementation, and the model may be published after the implementation is already complete.

A subtle but important distinction that may have significant bearing on the dependencies and constraints for consumers of the model.

Type		Basis	Service Model	Example
Service Model Implementation Ecosystem	Platform Specific Infrastructure	Adoption of platform ecosystem as foundation for SOA	ISM	J2EE Microsoft Windows
	Application	Adoption of application ecosystem as implementation of BSM	BSM	SAP Netweaver ESA
Service Model Ecosystem	Platform Independent Infrastructure	Adoption of common Infrastructure Service Model for Application interoperability (e.g. B2B)	ISM	OASIS ebXML WS-STAR
		Adoption of common Infrastructure Service Model for Resource Management	ISM	Grid OGSA
	Industry or Business Domain	Adoption of common Business Service Model for B2B interoperability	BSM	ACORD
		Extension of, and integration with application via Service Model	BSM	eBay Amazon
Service Framework Ecosystem	Solution Framework	Adoption of a framework comprising a mixture of platform, application components and Service Model tailored towards a specific solution or industry	ISM BSM	Microsoft Connected Services Framework

Table 2 – Service Ecosystems

Table 2 shows the types of Service-based ecosystems on a similar basis to Table 1. Again a framework that combines elements of platform and application, now together with a Service Model, is a further type of Service-based ecosystem.

The role of participants in Service-based ecosystems is primarily the same as with other IT ecosystems, to extend and use the components of the ecosystem.

3. Benefits and challenges with ecosystems for SOA

Ecosystem Benefits

General benefits of an IT ecosystem can be seen as

- Ecosystem wide standardization and commonality plus the support provided to participants by the ecosystem
 - Reduce time to participate in the ecosystem
 - Reduces the time to solutions for the ecosystem
- Participants in ecosystem provide extensions to broaden the coverage – such as increased functional, support on additional operating platforms. Hence, as the ecosystem grows, then participation grows and subsequently coverage grows even further.

However, the benefits of IT ecosystems may depend on the role of the participant as shown in Table 3. Whether the participant is a user organization or vendor the benefits and objectives are largely the same.

Participant	Benefits and Objectives
Dominant organization (user or vendor)	<ul style="list-style-type: none"> • Control over ecosystem • Ensure own objectives met • Lock-in other participants • Ensure support for new initiative • Increase market share • Exclude competitors
Other user organizations (user or vendor)	<ul style="list-style-type: none"> • Support of ecosystem • Ability to participate in ecosystem • Access to larger market • Compliance may be requirement to do business with dominant participant • Dominant organization support (financial, technical and marketing) • Favorable terms and conditions to members of ecosystem (e.g may be price benefits)
Developers	<ul style="list-style-type: none"> • Increased opportunity for developers • Ensure skills match market need

Table 3 - IT Ecosystem Benefits and Objectives

Service Model Ecosystems inherit these same benefits and objectives as those in Table 3. In addition, Table 4 lists further benefits and objectives for Service Model ecosystems and also adds additional roles of Service Provider and Consumer.

Participant	Benefits and Objectives
Dominant organization (user or vendor)	<ul style="list-style-type: none"> • Interoperability across the ecosystem • Make it easier for others to do business or work with them
Other organizations (user or vendor)	<ul style="list-style-type: none"> • New entrants only have to agree to protocols and semantics, not use of technology • Loose coupling minimizes dependencies • Loose coupling with protocol and message based approach makes it potentially easier to participate in multiple ecosystems • Trojan horse. Ease of entry into ecosystem enables them to then displace other participants – possibly to ultimately become dominant
Service Providers	<ul style="list-style-type: none"> • Higher number of Service Consumers who comply with same standards
Service Consumers	<ul style="list-style-type: none"> • Choice of Service Providers who comply with same standards

Table 4 - Further Benefits and Objectives of Service Model Ecosystems

Ecosystem Challenges

The prime challenge for participants is that ecosystems often lock them in as illustrated in Table 5. Though there may be no particular hard-wired technical or commercial dependency that ties them to the ecosystem, the cost of moving to another ecosystem at a later date is seen as prohibitive once solutions for the original ecosystem have been put in place and such a move is undertaken only if absolutely necessary.

Area		Challenge
Technical	Technology	Locks participants into use of specific platform or technology
	Implementation	Locks participants into use of specific implementation/application (as well as platform)
	Developer	Language and/or tool locks service implementation into platform
Business	Commercial	Entry into ecosystem locks participants into a commercial agreement
	Organizational	Participants will only do business with members of the ecosystem. Makes leaving the ecosystem difficult
	Geographic	Ecosystem may only operate on a geographic basis, creating challenge or overhead for international organizations

Table 5 - Ecosystem Challenges

Ecosystems often have a dominant organization (whether end user or vendor) who push adoption of the ecosystem primarily for their own objectives. There may be an objective to lock in participants. Whilst the user organization may not be the vendor of a specific technology, they may nevertheless use technology as a way of locking in participants by requiring its use across the ecosystem in the same way as a vendor might.

Ecosystems that are developed by industry consortium for the mutual benefits of participants may be seen as less threatening in this respect. However, whilst such initiatives that introduce standards that simplify interoperability across members of the ecosystem, such as

EDI standards, at the same time some of the strengths of an ecosystem may not be so apparent in that there may be less support given by participants to others, or there is no dominant participant driving adoption - hence usage might be seen as optional rather than mandatory.

Often it is not the standards promoted by the industry consortium that locks participants in. Rather it is the way in which participants develop solutions or the products they buy in support of those standards that inadvertently locks them in through lack of agility in their design.

However, as shown in Table 5 it should be recognized that technical and business are two separate areas of lock-in issues. Concern about technical lock-in may be irrelevant (even though still undesirable) if business constraints take precedent. That is, either

- the technical lock-in is accepted because business conditions demand participation, and use of technology demanded by the ecosystem (especially where there is a dominant participant) is obligatory
- even where there is no apparent technical lock-in created by platform or implementation dependencies, participants are still locked in to a particular implementation ecosystem by commercial contracts or constraints

Service Model Challenges

As already discussed, a prime reason to adopt SOA is to minimize dependencies across endpoints - consider endpoints also as participants in an ecosystem. Hence, SOA should be an ideal approach to reduce lock-in created by an ecosystem.

Though note though the previous point on business constraints taking precedent over technical concerns. SOA does not change this equation, though should prepare an organization to more easily move to other ecosystems when business conditions change.

However, whilst this is core SOA concept, in practice the Service Model may still be subject to some level of dependency. For example,

- it may expose dependencies on the underlying platform or implementation/application
- dependencies across components of the underlying platform or implementation may constrain flexibility in the Service Model
- Commercial contracts or copyright may bind the user of the Service Model to a specific implementation (or vendor or provider), or vice versa

These are considered further in Table 6.

The Role of Ecosystems for SOA

Type	Potential Constraint	Impact
Service Model Implementation Ecosystem		
Platform Specific Infrastructure	The Service Model (BSM or ISM) is proprietary to, or creates dependencies to the platform ecosystem i.e. exposes behavior that is specific to the platform	Unable to switch to alternative platforms
	Service model is dependent on behavior offered by specific middleware platform. E.g. Reliable Messaging must be provided by the transport layer	Unable to use alternative transports or middleware
	Code to implement Services is specific to the platform. Even if the language is portable, the use of class libraries and behavior offered by the platform constrain the application to that platform	Whilst the Service Model might have no dependencies and be implementable on many platforms, the actual implementations themselves are not portable.
Application	BSM is proprietary to an Application, or the Application ecosystem e.g. copyright or commercial constraints prevent the BSM being used with other applications, or component of	The Service Provider is locked in to the Application Service consumers are indirectly locked in to the application by use of the Service. However, other there still may be alternative providers of the same Service who themselves use the same application
	BSM is not sufficiently abstracted from the Application to hide implementation e.g. the schema is a direct reflection of the application's interfaces and/or data model	Use of a Service creates dependence on implementation. E.g. may depend on data values that are only supported by the implementation
	The application creates dependencies across Services in the BSM, and/or across the Service Ecosystem The application or application ecosystem creates dependencies across Services in other business domains, depending on the scope of the application e.g. A large monolithic ERP might introduce dependencies across many Services	Lack of flexibility at a Service level. Alternative Services from other providers cannot be used
	The BSM is not a standard, or doesn't follow standards, within the business domain.	Cannot easily use alternative Services that use different BSM
Service Model Ecosystem		

The Role of Ecosystems for SOA

Type	Potential Constraint	Impact
Platform Independent Infrastructure	<p>May not have the full backing of platform vendors</p> <p>Lack of choice of available implementations (support of Service Model by vendors)</p>	<p>Whilst the Service Model design creates no technical dependencies on the implementation, the lack of choice in reality creates a constraint</p>
Industry or Business Domain	<p>Lack of choice of available implementations (support of Service Model by application vendors)</p>	<p>Whilst the Service Model design creates no technical dependencies on the implementation, the lack of choice in reality creates a constraint</p>
	<p>Service Model may not be recognized as a standard within the business domain (de jure or de facto). Proprietary just to ecosystem. May be multiple competing Service Models and ecosystems in that domain</p>	<p>Participants are forced to support multiple Service models across different ecosystems</p>
Service Framework Ecosystem		
Solution Framework	<p>Dependencies at the application and platform level impact Service model flexibility</p> <p>There may be a commercial constraint that prevents "mixing and matching" of framework components from different providers</p>	<p>Though there are no apparent dependencies between the individual Services and the corresponding underlying application and platform, dependencies between application components, and/or platform create dependencies across Services</p>

Table 6 - Potential Service Model Constraints

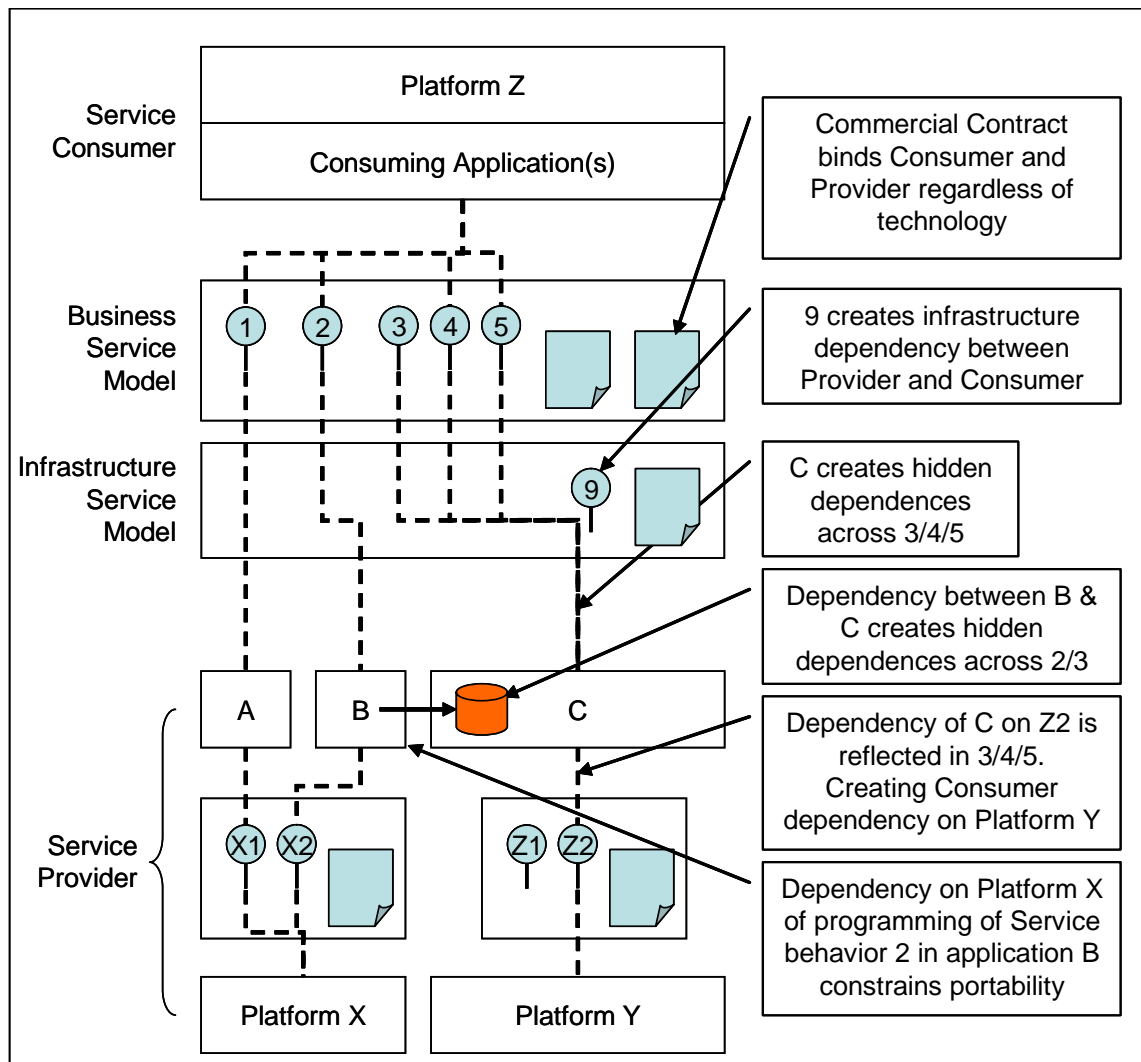


Figure 2 - Dependencies in SOA

Figure 2 illustrates these potential dependencies across the model layers. In this,

- Service 5 cannot be provided independently of Service 4. For example, database dependencies within C prevent 4 and 5 being provided by separate implementations (without those database dependencies being recreated in those implementation).
- Similarly, Service 2 cannot be used independent of 3 because of the hidden dependency between B and C
- Services 3/4/5 require behavior (such as Reliable Messaging) that is provided by the middleware in the underlying platform Z that hosts C. Consequently the Service Consumer must also use Z in order to meet the behavioral requirements of 3/4/5
- The code to support the behavior of Service 2 is described in program B using capabilities in Platform X. Whilst the Service specification or behavior of 2 might show no dependencies on X, and could easily be replicated on a different platform, the code in program B is dependent on X and is not portable
- The infrastructure protocols in the ISM that is used by the BSM (such as use of Web Service Protocols) binds the Provider and Consumer to use the same protocols. Similarly, Service 9 in the ISM, (for example a Registry Service) creates a

dependency between Provider and Consumer to use the same infrastructure Service.

The potential consequences in an ecosystem for SOA are

- Inability to dynamical resolve providers and endpoints. Dependencies on a specific instance of a Service constrain the ability of the Service Consumer to dynamically resolve endpoints or to choose between Service Providers.
 - This might be an objective for a dominant provider who wants to constrain choice in the ecosystem.
- Lack of resource virtualization and distributed implementation architecture. Dependencies in the application and platform prevent SOA from helping to deliver a scalable, resilient distributed or federated architecture. Service Providers cannot easily virtualize their resources, or components of it.
 - This might be an objective of a dominant vendor in an ecosystem who does not have support for distributed or federated architecture, or does not users potentially hosting components on different platforms.
- Inability to federate Services across the ecosystem. Service provision may become centralized, or dependent on certain Providers as dependencies prevent other participants in the ecosystem easily becoming hosts.
 - This may be an objective of a dominant participant – to constrain others ability to compete as providers.
- Inability to integrate across ecosystems. Dependencies may make it difficult for a Service Provider or Consumer to operate across multiple ecosystems without duplication of resources and/or effort.
 - This may be an objective of an ecosystem that wants to constrain growth in competitive ecosystems.
- Inability to host Service Implementations on alternative platforms. Dependencies in the code and/or development tool are specific to the platform
 - This may be the objective of a vendor who wants to lock service implementations to their platform
- Inability to use a Service Model as an agility layer. Commercial contract-based dependencies created between the Service Model and the Implementation may make it impossible to use the Service Model as a mechanism to decouple Service Consumers from a specific implementation and hence enable subsequent replacement of the implementation with an alternative with minimum impact on the consumers. Whilst technically the Service Model might enable this, a commercial contract could prevent it.
 - This might be the objective of a packaged application vendor who wants to lock in Service Consumers to their implementation

Appendix I provides a checklist for Service Ecosystems and Service Models to assess these constraints.

Summary

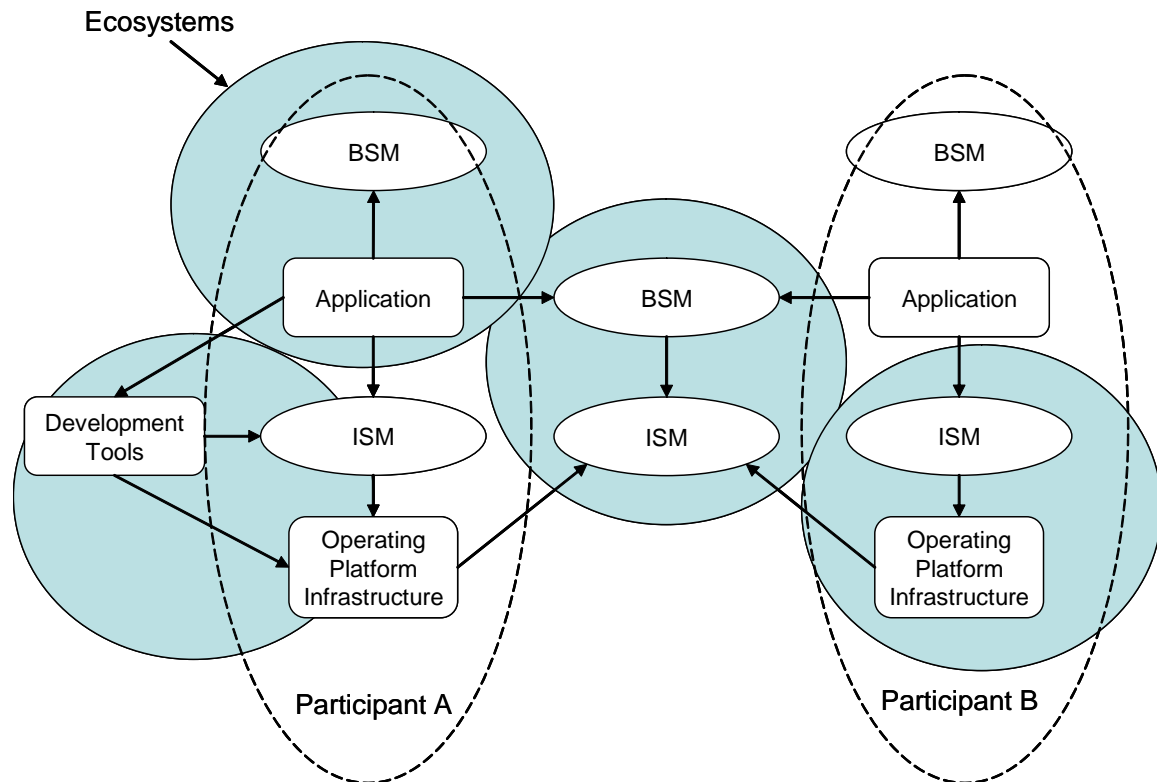


Figure 3 - Dependencies and Ecosystems

Figure 3 summarizes the dependencies between different elements of the SOA and where ecosystems form to support SOA.

Dependencies can be seen between

- The applications of each participant are dependent on the shared BSM for the Services and semantics of their message exchanges. For example in a B2B context.
- In turn, they and the BSM are dependent on the ISM that defines the infrastructure protocols and services they must use to enable the business exchanges.
- This ISM is then dependent on the operating infrastructure that supports those protocols and Services.
- The applications may also be dependent on the BSM for that business domain – which might have a broader model than the BSM that is specific just to the needs of B2B.
- The application is also dependent on the ISM that binds it to the underlying operating infrastructure.
- Finally, the application is also dependent on the development tools used to build it, which in turn are also dependent on the ISM for the operational infrastructure, and are also typically tightly bound to a specific operational infrastructure in more direct ways than via a Service Model.

Ecosystems are formed around

- The application BSM and its implementation

- The BSM shared between the participants
- The ISM and its implementation in the operational infrastructure
- The development tools for an ISM and the operational infrastructure

4. Key Ecosystems for SOA

The following is not an exhaustive list of ecosystems for SOA and Web Services, but are representative of the state of the art.

Business Service Model ecosystems

The provision of Business Service Models that are specifications of Services defined independently of their implementation is a relatively new idea, but one that can be seen as an evolution of initiatives that provide document/message schemas or data model as a basis for interoperability, or object models that provide a starting point for building an implementation.

As such many BSM will come from existing industry initiatives or vendor activities that evolve their specification to support SOA. This is not to imply this is always the ideal route in terms of creating the best Service Model, but is nevertheless the expedient route that many initiatives will take in the hope it eases the transition of current users to SOA.

Initiatives that have focused primarily on the delivery of document schemas and data model may provide the foundation for interoperability but do not provide a complete Service Model on their own. The Service Model also needs a description of the behavior required (such as a process model) for each message exchange so that the individual Service Operations can also be specified. As stated earlier, in order to for both participants to connect their applications using SOA, requires a mutual understanding of the semantics, protocols, schemas, behavior and process involved.

Industry Initiatives

Many industry initiatives for document and process standards have primarily evolved out of the EDI era. Though many will have adopted the use of XML for their document schemas, most have not yet moved to support Web Service based messaging. As such, whilst they provide key elements they do not yet provide rich enough Service Models that would fully address the needs of SOA. Examples include the following:

ACORD. A leading example of an industry initiative for SOA is ACORD² in the insurance industry. They have updated their specifications to support XML documents and the use of Web Service based messaging. The ACORD XML for Life and Annuity Messaging Service provides for the use of SOAP and WSDL – though ACORD can use other messaging frameworks too. It leverages the ACORD Life and Annuity Specification which provides a full set of data types and objects to provide semantic interoperability, and defines transactions (operations) and their choreography to support the business processes.

As such, ACORD XML provides a sound basis for SOA. As an industry initiative it has global recognition and support.

IBM has recently contributed its own Insurance Application Architecture (IAA) to ACORD

HL7. Health Level 7³ for the healthcare industry is an ISO standard. HL7 has been rebuilt in V3 to provide a proper information model and the use of XML. Both Web Services and ebXML are currently supported by the HL7 V3 Messaging Standard for “trial use”. This

provides for more robust standards-based messaging behavior with the use of reliable messaging, routing and security.

The Benefits and Challenges of using industry Service Model initiatives as the basis for SOA include

- Benefits
 - Best industry initiatives will have industry-wide use and support. Wide-scale usage by end-users encourages vendor support and the development of an ecosystem of vendors providing rich capability for that market. A large market also potentially brings commoditization and reduction in costs.
 - True industry initiatives should have no dependency or lock-in to platforms or implementation. This is not to say that some vendors might have an unfair advantage because they contributed to the initiative by providing IP which is already in their products.
 - Should be open and usage should be royalty free.
- Challenges
 - Incomplete Service Model specification. May not be rich enough to provide a complete specification of the Service model. Parts of the specification may only be defined in the implementation detail, rather than as an abstract specification. Parts of the specification such as policy and contracts may be open to interpretation by users.
 - Lock-in to the ecosystem. Use of the standard is a pre-req for participation in the industry, or a pre-req for doing business with participants in an ecosystem. Fine when there is only one standard, but a constraint if there is not, or the standard is immature/incomplete.
 - Barrier to entry. Organizations need to be careful where enforcing compliance with the standard. Compliance across the industry should be seen as a way to lower the cost of entry due to simplified integration and commoditized solutions, and not as a barrier to entry through over-complex specifications, unnecessary pre-requisites of dependencies, and subsequent high cost of compliance.
 - Only basic support for Web Services. With some key Web Services protocols not yet approved as standards it is not surprising that many industry initiatives do not yet specify their use. Consequently, some key aspects of the service behavior might use a proprietary approach, for example reliable messaging headers. This is unlikely to be resolved until the Web Service protocols themselves are approved as standards. For example there is little use BPEL for process/service orchestration, or WS-Security for authentication/encryption.
 - Industry initiatives often take longer to evolve than vendor-led activities. The need to get consensus across the industry is much more time consuming. Hence the delay in Web Services support. On the positive side, standards built by consensus should be readily adopted and more durable
- Solutions
 - Users need to understand where the gaps are in the Service Model and work to fill these in collaboration with other participants in the ecosystem – at least till they become part of the specification

Application Vendor Initiatives

Application Vendor Business Service Model initiatives are far more likely to have dependencies on that vendor's technologies or products. This might be a technical constraint, but is more likely to simply be a commercial one, in that the Service Model can only be used in conjunction with their implementation of the Services.

SAP ESA. SAP Enterprise Services Architecture⁴ defines a high-level Service Model that spans the core business processes of most organizations. It is the most richly developed SOA from a packaged application vendor, and is not just limited to SAP in that it describes abstract Business Services that are not just a reflection of the underlying SAP interfaces (BAPI). The use of SAP Netweaver Middleware enables the use of different implementations, for example aggregating information from multiple sources including existing in-house systems or other packaged applications. However, commercial contracts restrict use of the Service Model to SAP users.

Though the Business Services themselves are well formed and abstract, their subsequent implementation using SAP can introduce dependencies across Services because of the underlying dependencies contained within the SAP application as explained in Figure 2.

eBay and Amazon. Though not often thought of as packaged application vendors, both eBay⁵ and Amazon⁶ provide access to their applications for use by other retailers, and by other vendors who wish to become part of their ecosystem by providing extensions and solutions based on the eBay and Amazon platforms.

Both have attracted a large ecosystem of developers and users who have extended the platform by building front-ends that provide enhanced e-commerce capabilities. They are also used by large retailers as an engine for their own e-commerce requirements rather than building their own in-house, or as a complement to their in-house systems.

Both eBay and Amazon are web-based and only accessible by 3rd parties via their published Services. Though both are often referred to as Web Services, there is minimal use of Web Service protocols, and they are not a pre-requisite. The Services can be used simply using HTTP and XML. Their Service Models are very specific to the processes in their applications. As such they are not generalized, abstract Business Services for e-commerce and not easily reused in other scenarios. As such, the specification assumes you already know the processes within their applications.

The Benefits and Challenges of using vendor Service Model initiatives as the basis for SOA include

- Benefits
 - Provide ready made Service models to access packaged applications.
 - Enable different degrees of loose coupling between the Service Consumer and the application. This might simply be technical loose coupling (no platform dependencies), or by use of well formed Business Services providing full loose coupling to the application implementation
 - Access to the Services has encouraged large ecosystems who
 - extend the application to meet end-user requirements
 - provide additional tools to simplify development of solutions based on the Services

- Challenges
 - Having invested in providing a Service Model to enable access to their applications, vendors are not likely to allow that same Service Model to then be used as a mechanism to switch users to a different application from a competing vendor
 - Commercial or copyright constraints limit usage of the Service Model to the vendor's application
 - The Service Model specification may not be complete on the basis that users will already be familiar with behavior of the underlying application
 - The Service Model may be highly specific to the behavior of the underlying application
- Solutions
 - Users should consider the using their own Service Model in conjunction with the vendor's Service Model (effectively another layer) and publishing those Services for Service Consumers. In this way, consuming applications will be bound to the organization's own Service model and not one that carries restrictive usage rights.

Infrastructure/Technology ecosystems for SOA

Infrastructure Service Models should provide access to the infrastructure requirements of the SOA without creating dependencies on specific infrastructure technology or implementations.

Some of this requirement is inherent in the protocols used – e.g. the use of WS-RX enables reliable messaging without requiring implementation of a common reliable messaging technology across all endpoints.

Other requirements should be met by the underlying infrastructure itself providing a Service Model to provide access to its capabilities rather than using proprietary APIs or requiring for example that all messages are processed by it.

ebXML and WS-Protocols both define a set of protocols for describing the infrastructure behavior required by messaging in SOA. Both define protocols for reliable, secure messaging (ebMS and WS-STAR). Additionally both define standard for a Service Registry, a key component of SOA infrastructure.

Though there are some protocols that are common to these specifications (SOAP, WSDL), use of either of these protocol sets currently locks the endpoints to those protocols. It is expected that both initiatives will converge in the near term, so that the specifications are complementary rather than competing. A detailed assessment of these specifications is detailed in a further CBDI Forum report⁷.

Grid OGSA. The Open Grid Services Architecture⁸ provides an Infrastructure Service Model that enables life-cycle management of virtualized resources (provisioned, purposed, monitored, etc) without creating dependencies on any specific platform, middleware or systems management product. OGSA defines eight high-level categories of services:

- Infrastructure Services - communication between disparate resources (computer, storage, applications, etc.)

- Resource Management Services - monitoring, reservation, deployment, and configuration of grid resources based on QoS
- Data Services – provide the movement of data to where it is required with replication, query execution and updates, and data transformation
- Context Services - describe the required resources and usage policies for each customer that utilizes the grid
- Information Services - provide information about the grid and its resources such as status and availability
- Self-Management Services - support the attainment of stated levels of service through autonomies
- Security Services - enforce federated security policies
- Execution Management Services – enable the workflow of grid processes such as provisioning, and lifecycle management

OSGA leverages Web Service protocols such as WS-STAR, WSRF (Resource Framework) and WSDM (Distributed Management).

OSGA should provide a useful ISM to support any type of application domain. However, to date it has been primarily applied in the scientific and technical computing domains. There has been some use in the financial services industry, but again this has been primarily to support high performance “number crunching” requirements.

OSGA provides an open standards based alternative to cluster-based computing that is more often provided by platform OS and hardware vendors. Perhaps because this is a competitive threat to these vendors – theoretically enabling users to virtualize resources across a heterogeneous environment – it has been less actively promoted by vendors. Development and leadership has come from the grid community itself, though vendors such as IBM have made significant contribution to the development of standards.

Whilst OSGA implementations can run on top of common platforms, ideally the platforms themselves would support OSGA natively. There is currently no indication that major platform vendors plan to do this. However, platform vendors should support the underlying base Web Service protocols used within OSGA as referenced above.

OSGA is not complete however in addressing all the infrastructure needs for SOA. Its prime focus is on resource virtualization. OSGA needs to be combined with an Enterprise Service Bus (ESB) to support the service/message mediation requirements of SOA.

CBDI Forum believes that grid OSGA can play a significant role in SOA. The Business Service Model provides business agility, whilst a well formed Infrastructure Service Model containing OSGA provides technical/implementation agility.

Vendor Infrastructure Service Models

Platform and infrastructure vendors have been slow to expose their own infrastructure resources as true Service Models. Many are configured and used via proprietary interfaces and configuration files. Whilst vendors promote the benefits of loose coupling etc., in applying SOA to business applications, few have applied SOA to their own infrastructure. Hence, whilst business resources might be more agile as a result of SOA, they are often still dependent on tightly coupled infrastructure resources.

This might not be seen as a problem today for most organizations who are looking for business agility to support the more important needs of the business. However, as IT organizations become more comfortable with the concepts of resource virtualization as they apply it to their business resources they will begin to recognize the need to virtualize infrastructure resources in the same manner.

The Benefits and Challenges of using infrastructure Service Model initiatives as the basis for SOA include

- Benefits
 - Virtualization and Loose Coupling. Service-based approach should decouple the use of infrastructure capability required by applications from the actual instances of infrastructure products
 - Provide a standards-based infrastructure platform. The provision of standards-based infrastructure models such as OGSA and the Web Services protocols it uses provide a viable alternative to the proprietary technologies used by platforms and middleware infrastructure today
 - Permit the orchestration of both business and infrastructure Services within a single process
 - Broad support for WS-STAR protocols
- Challenges
 - The lack of Infrastructure Service Models currently provided by the major platform vendors
 - Grid currently faces a major challenge in gaining market acceptance for business applications
 - Industry needs to converge ebXML and WS-STAR

Application Servers/Containers and Programming Models for SOA and Web Services

J2EE and Microsoft Windows. Both J2EE⁹ and Microsoft Windows .NET¹⁰ are application containers that provide programming models that simplify the delivery of Web Services. Both enable the developer to build applications that provide and consume Web Services without having to code much (or any) of the XML that is required. The developer programs in their chosen language against a framework of class libraries that hide the detail of Web Services protocols and behavior.

Both can also target other technologies that might be suitable for some styles of SOA, such as the use of Message Oriented Middleware (MOM) or Object Request Brokers (ORB) though these may be best applied in different scenarios to Web Services. For example, when all the endpoints are within the same organizational boundary.

Specific J2EE containers such as IBM WebSphere or BEA WebLogic may build upon the base J2EE libraries to provide additional support for generation and delivery of Web Services. For example supporting emerging Web Service protocols that have not become part of the J2EE standard (such as WS-RX for reliable messaging)

Extensive ecosystems have developed around both J2EE and Microsoft Windows .NET to provide extensions, tools and solutions based on their programming model that help to make developers more productive in their use.

J2EE, Windows and Apache (see below) Application Servers also provide run-time support for SOA by providing APIs that enable other tools (and in-house applications) to interrogate and manipulate the messages as they flow through the server. This has seen the growth of ecosystems of Service Management and ESB tools that support these Application Servers and can for example enforce Service policies, or route Service requests.

IBM CICS. IBM has developed CICS to become a first class provider and consumer of Web Services. That is, it understands Web Service protocols natively and requires no additional wrapper or integration layer. CICS programmers are able to use conventional CICS commands and let the CICS container deal with the details of the Web Service protocols and behavior.

IBM CICS is one of the few long standing application containers to have been upgraded in this way. Others typically rely on some wrapper or external integration layer to provide support for Web Services.

LAMP. LAMP is an abbreviation for the popular mix of Open Source Software (OSS) based on Linux operating system, Apache Web server, MySQL database and Perl or PHP scripting language that is used to build many websites. The Apache Web Services project¹¹ is an initiative to provide support for Web Service protocols in Apache – typically based on corresponding Java initiatives.

The Benefits and Challenges of using application Containers and programming models as the basis for SOA include

- Benefits:
 - Developer productivity
 - hide the complexity of Web Service protocols
 - hide the complexity of Web Service behavior
 - Reduce training effort, reuse existing skills. Developer continues to use chosen programming language
 - Can help to minimize impact of changes in Web Service protocol specifications
 - Extensible. New protocols or behaviors can be added
- Challenges
 - Coding for the behavior and the protocols required is container specific. A J2EE Web Services application cannot be easily moved to Microsoft .Net even though it supports the Java Language or vice versa. Therefore Service implementations may not be portable
 - Coding for the behavior and the protocols required may be messaging/connectivity technology specific. For example. The code to use Web Services as the implementation for SOA, may be different to the code to use MOM.
 - Behavior may not be totally compatible across different containers. That is, even though the same protocols and behavior have been specified, there still may be interoperability issues between different containers. For example a Microsoft .NET client consuming a J2EE provider.

- Using container specific extensions to J2EE to support emerging Web Service protocols may lead to interoperability issues, and possible “dead ends” if those protocols fail to emerge as standards
- Solutions
 - Interoperability challenges can be partially overcome by conforming to interoperability profiles published by the Web Services Interoperability Organization (WS-I). Containers that generate Web Service code that comply with these profiles will be interoperable with others that conform. However, these are currently limited to base Web Services protocols. Though it is fair to say it is difficult for them to publish profiles for protocols that are themselves not yet finalized as standards.
 - Microsoft is addressing the challenge of code being messaging/connectivity technology specific with the introduction of the Windows Communications Framework (WCF). This further abstracts the specification of messaging behavior from the protocols and technology used. Behavior is specified in the same way for all supported protocols and technology, and policies then determine the actual protocols and technologies to be used at runtime. In this way, the protocols and technologies can be changed by changing the policy (and XML configuration file) without impacting the code. Extensibility enables additional protocols and technologies, including support from 3rd parties.
 - Developers should separate Service logic from business logic. Coding the two inline, particularly when the level of abstraction of the Service logic is not high, or dependencies on the platform are obvious, as this might constrain agility and increase maintenance effort. Service logic should be separated so that it can evolve independently of the business logic.

5. CBDI Recommendations

In view of the above CBDI makes the following recommendations

1. Organizations first need to decide where their Service Models come from, and hence which ecosystems they participate in. They may be

- Developed in-house
- Based on industry initiatives
- Acquired from a vendor – typically along with their products

This decision on the source of a Service Model will be based on factors such as

- What is the effort involved?
- Will they be compatible with the Service Models of other participants?
- Should the Service Model be developed in collaboration with other participants
- Are there existing Service Models available for each business or technical domain?
- What are the constraints in using a Service Model developed externally, such as from a vendor?
- Does external Service Models that are used or acquired comply with open standards?
- What is the availability of implementations to support the Service Model?

Organizations should assess available Service Ecosystems and their Service Models as per Appendix I.

Organizations will most likely adopt a mixture of all three sources of Service Models as

- Other reasons for acquiring a vendor's products override concerns about constraints or dependencies in using their Service Model
- The business ecosystem they participate in, or a dominant business partner, demands use of their Service Model
- The lack of availability of suitable Service models for a particular business domain

2. Organizations should adopt strategies for reducing dependencies and constraints in SOA ecosystems

Obvious strategies would be to only use or acquire Service Models that are based on open industry standards, and only enter in Service Ecosystems on that basis.

User organizations should define their own Business Service Models. Figure 4 and Table 7 illustrated a layered architecture for the different Service types that may be found in the Business Service Model. In this,

- Only Published Services in each layer can be consumed by the layer above. That is, other Services might exist within that layer, but these are only used within that layer.
- Published Services should be at an appropriate level of abstraction so they do not expose dependencies on lower layers in the architecture
- Identify and publish their own core Business Services and Process Services. These can be based on open industry initiatives where applicable, and where any constraints are acceptable
- Publish Services in Vendor Service Model only where constraints and dependencies are acceptable— or is compliant with industry initiative.
- Otherwise, access to the Vendor Service model should only be via published core Business Services and Process Services. Figure 4 illustrates that the organization's core Business Services are mapped to the vendor's Business Services. Dependencies from the solution layer or Process Services above are always made via the organization's Business Services.

Figure 4 might appear to suggest an overly complex model. However, organizations only need to implement this strategy for those Services that might otherwise introduce unwanted dependencies or constraints on Service Consumers if they were to use them directly. Where a Service in the Vendor Service model places no such constraints or dependencies that might inhibit future agility, then it can be used directly.

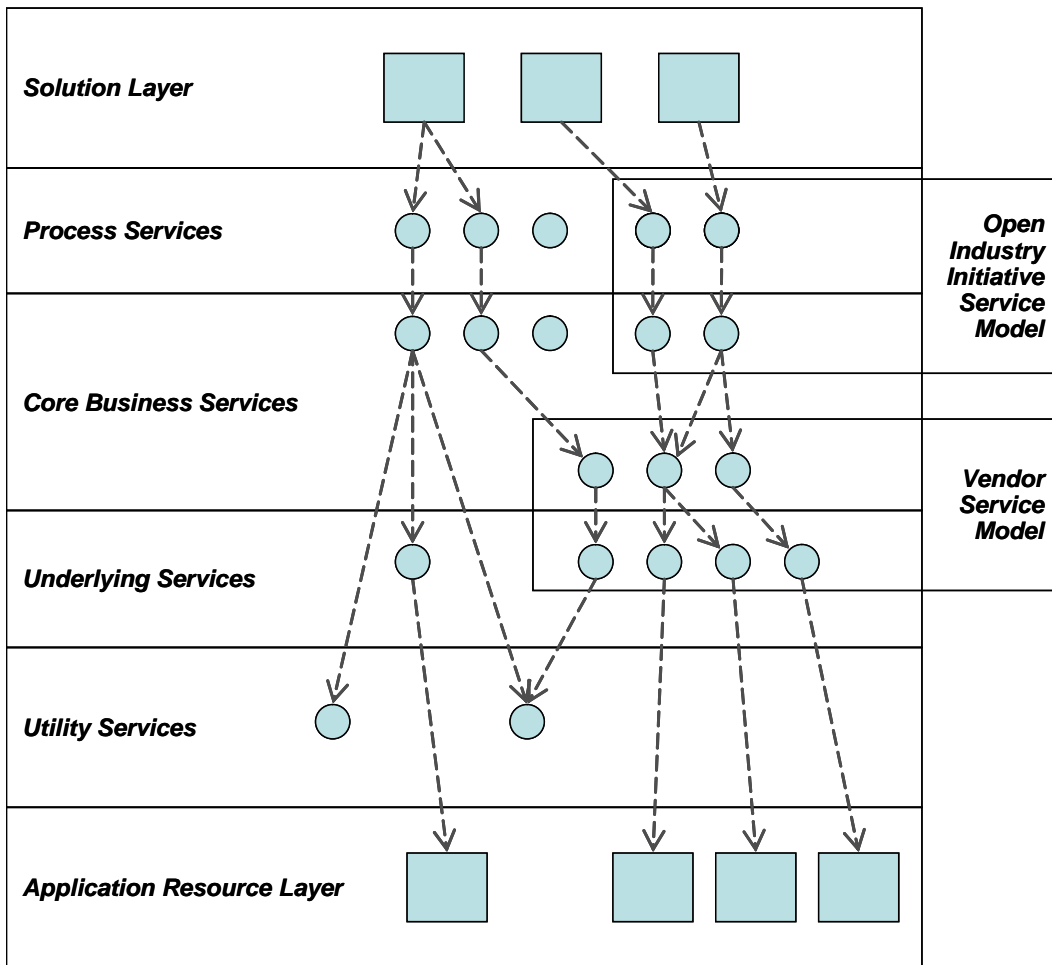


Figure 4 - Service Model Layers

Organizations should always undertake to identify their own Business Service Model in order to understand their particular requirements and to form a basis against which to map industry initiatives and vendor models when selecting possible candidates for adoption.

It is beyond the scope of this report to show the process for identifying the BSM. CBDI Forum has authored methodology guidance for this which is published in the CBDI Journal¹²

In some domains the industry initiative or vendor model might be a self-selecting choice for the Business Service Model as it is a pre-requisite for entry into an ecosystem, as explained earlier in the report

Layer		Source
Solution Layer	Service Consumer	
Process Services	Expose business process steps to Service Consumer (solution layer) These will be the meaningful to the Service consumer - that facilitate some business activity between participants	Based on Industry Initiative where appropriate Define own Process Services
Core Business Services	Core Business Services identified from core Business Type model (things of meaning to the business)	Define own Core Business Service model Use Industry Initiative where Open or acceptable constraints
Vendor Service Model	Might contain a mixture of process, core and underlying Services	Vendor
Underlying Services	Services exposed by the underlying implementation	Vendor Service Model – underlying Services on their implementation Existing system interfaces Other External Services from external providers
Utility Services	Common utilities	various
Application Resources	Implementation of the capability provided by the Service	Packaged Applications Existing Systems

Table 7 - Service Model Layers

3. In Service Design and Implementation, organizations should use the following approaches.

Separate Service Logic from Business Logic - Modern programming environments for SOA such as Microsoft Windows .NET and Visual Studio make it easy to code Service Logic. Emerging technology such as Windows Communications Framework allows it to be expressed even more easily in a simple declarative manner. Consequently, there is a danger that Service Logic will be mixed with Business Logic. Developers should ensure that they continue to use best practice of separating presentation, business rules and data, where the Service Logic is contained in its own layer.

Organizations might take this a step further and consider hosting the Services and delivering the Service Behavior outside of the application container. For example using an ESB to host the Services, and connect those Services to the underlying implementation

In this way,

- changes to the Service behavior are isolated from business logic
- changes to the Service platform are isolated – for example, migrating from use of Windows WSE (Web Services Extensions) to the use of Windows WCF may require significant maintenance of existing code. This would be simplified if that code was isolated.
- Service logic code can be reused across many business components

Similarly, Service Policies should always be managed outside of the business logic. Organizations should consider ways in which Service Policies can be managed centrally and can be easily changed at runtime without maintaining code. Service Management tools and ESB products typically provide such capabilities. The platform itself may provide the APIs to the message streams as mentioned earlier in which policies can be enforced - Service Management tools and ESB products often take advantage of these exits

Use a “Contract First” approach. - That is, the Service Specification is always developed before the implementation is built. The specification should include the contracts or obligations (pre-post conditions) on both provider and consumer. The implementation is then built to meet the specification and honor the contracts and obligations.

4. The role of Government and Public Sector in Service ecosystems should be to

- Set a high standard for levels of abstraction, technology/implementation independence etc in their development or acquisition of Service Models to ensure no bias is shown towards a particular technology, product or vendor
 - Some Service Models – particularly Infrastructure Service models – are encapsulated in products and technology and are therefore part of their acquisition and usage. As mentioned earlier, other conditions that affect the choice of products or technology might override the selection of a Service Model. Government and Public Sector is no different from commercial organizations in this respect. Government and Public Sector still need to take expedient and cost effective routes to solutions. Guidance in 5.3 above should therefore be applied.
- Encourage adoption of open industry standards
- Focus on establishing Business Service Models for their domain (whether, B2G, B2G, C2G, G2G), and work collaboratively with participants in those domains to ensure compatibility
- Take a leadership position in standards adoption. Set an example to encourage adoption by businesses. In our experience, many businesses are playing a “waiting game” on standards and looking for leadership, or only adopt them when absolutely necessary.
- Participate in standards activities to ensure the needs of government and the public sector are recognized, not just business and vendors
- In the climate of global business, governments should foster International co-operation on SOA standards and ecosystem development through initiatives at EU, UN and other appropriate levels

Appendix I: Service Ecosystem Assessment

Table 8 provides a checklist for making an assessment of a service ecosystem

This is split into four areas

- **Service Model.** Assessment of the Service Model to understand dependencies it might have on the implementation (application or platform) or factors that might constrain future agility. This might be a factor of poor design, or deliberate move by the provider of the Service Model to lock participants into their ecosystem. The more abstract the Service Model, and the fewer dependencies or constraints, the more flexible it is likely to be in supporting future requirements.
- **Technical.** Assessment of the technical behavior of the ecosystem and the implementation to understand again whether this creates hidden dependencies within the Service Model and other components in the implementation ecosystem.
- **Commercial/Business.** Assessment of the commercial and business factors that might create dependencies and/or constraints within the Service Ecosystem
- **Usability/Agility.** Finally, the previous three areas of assessment can used to determine the usability and agility provided by the Service Model. Does it provide choice or flexibility for Service Providers or Consumers?

Area	Assessments
Service Model (service specifications)	
Richness of the Service Model specification	<ul style="list-style-type: none"> • Asses which specifications are provided as part of the Service model. Including <ul style="list-style-type: none"> ○ Service Operation specification ○ Document/message schemas ○ Taxonomy/ontology for semantics ○ Process models ○ Service information model (data model for the services) ○ Policies ○ Contracts
Level of Abstraction of Services from implementation (application dependencies)	<ul style="list-style-type: none"> • Degree to which Service hides Implementation specific behavior or data. E.g. no codified data • Whether there is full use of XML tags to mark up data, or use of binary data or strings • How well Services reflect recognizable Business functions (or function of the domain) – that is meaningful to the Service Consumer, and are not just a representation of an interface to the existing implementation
Level of abstraction of Service Model from implementation (platform, infrastructure and technology dependencies)	<ul style="list-style-type: none"> • Degree to which Service hides platform specific behavior or data. E.g. no dependence on platform specific functionality – see Technical behavior • Check proper separation of Business Service model from Infrastructure Service Model
Compliance with standards	(de jure or de facto)
<ul style="list-style-type: none"> • Business Semantics and Process 	Test compliance with relative vertical industry domain standards for Business Service Model

The Role of Ecosystems for SOA

Area	Assessments
<ul style="list-style-type: none"> • Technical 	<ul style="list-style-type: none"> • Compliance with open standards. E.g. <ul style="list-style-type: none"> ○ Web Service Protocol Compliance ○ ebXML compliance • Test compliance with industry domain standards for Infrastructure Service Model (e.g. Grid OGSA)
Service Model specification level of detail	<ul style="list-style-type: none"> • Level to which Service is fully Specification • Whether necessary to inspect implementation to understand behavior
Technical	
Technical Behavior of Services	<ul style="list-style-type: none"> • Degree to which Service behavior is independent of platform ecosystem, such as <ul style="list-style-type: none"> ○ Transactions ○ Reliable Messaging ○ Security • Extent of use of standards-based protocols to convey behavior rather than dependence on platform/transport (e.g use of WS-RX rather than MOM for reliable messaging)
Implementation-level dependencies	<ul style="list-style-type: none"> • Extent of dependencies across Service Model created by dependencies in implementation
Development tool dependencies	<ul style="list-style-type: none"> • Extent to which Service Model specifications and behavior are described only in development tool • Whether Service Model is held in a proprietary format • Whether full Service Model is machine readable by other development tools, or held as declarative information by tool
Componentization of implementation	<ul style="list-style-type: none"> • Assess granularity of implementation (as an inhibitor to Service Model scalability and resilience) • Whether there are non-service based dependencies between components (e.g. direct access to database across components) that constrain agility, or introduce Service Model dependencies
Commercial/Business	
Service ecosystem contracts	Whether participation in ecosystem carries contractual obligations
Service Model usage contracts	Whether use of Service Model is contractually bound to use of a specific implementation
Provider contracts	<ul style="list-style-type: none"> • Whether the Service Consumer is contractually bound to provider • Whether use of Service Model is contractually bound to a specific provider
Copyright	Whether Service Model is copyright of <ul style="list-style-type: none"> ○ Provider ○ Vendor ○ Industry consortium Assess what restrictions copyright places on use of Service Model

The Role of Ecosystems for SOA

Area	Assessments
Regulatory compliance	Whether Service Model and/or implementation complies with relevant legal regulations
Usability/Agility	
Service Consumer Choice	Assess the range of Service Providers in the ecosystem. Does the ecosystem provide choice of providers (runtime or design time) sufficient to meet the needs of the Consumer? Are limitations in choice acceptable?
Service Provider Choice	Assess the range of Service Implementations in the ecosystem. Does the ecosystem provide choice of implementations (application or platform) sufficient to meet the needs of the Provider? Are limitations in choice acceptable?
Service Consumer Flexibility	Assess the ability to change Service Providers. Does use of the Service Model constrain the ability to switch Service Providers?
Service Provider Flexibility	Assess the ability to change implementations. Does use of the Service Model constrain the ability to switch Service Implementations?
Ease of Migration	Assess the degree to which the Service Model enables migration to alternative implementations. Does the Service Model provide sufficient loose coupling such that it enables future migration to alternative implementations and/or providers?
Market	
Use by other Participants	<ul style="list-style-type: none"> • Assess participants in the ecosystem. <ul style="list-style-type: none"> ○ Is there wide participation from business partners or other relative organizations ○ Is the Service Model actually used widely by participants?
Development	<ul style="list-style-type: none"> • Should the Service Model be developed in collaboration with other participants
Availability	<ul style="list-style-type: none"> • Assess which Service Models are available for the domain • Assess the extent to which implementations of the Service Model are <ul style="list-style-type: none"> ○ Widely available ○ widely used ○ Cost effective

Table 8 - Service Ecosystem Assessment Checklist

About CBDI

CBDI Forum is an independent industry analyst and consultancy company. CBDI Forum provides a focus for the industry on best practice in business software creation, reuse and management, specializing in Service- and Component-based approaches including Service Oriented Architecture (SOA).

The information available in CBDI publications and services, irrespective of delivery channel or media is given in good faith and is believed to be reliable. CBDI Forum Limited expressly excludes any representation or warranty (express or implied) about the suitability of materials for any particular purpose and excludes to the fullest extent possible any liability in contract, tort or howsoever for implementation of, or reliance upon, the information provided. All trademarks and copyrights are recognized and acknowledged.

<http://www.cbdiforum.com>

Lawrence Wilkes is a Principal Analyst of CBDI Forum.

Contact at lawrence.wilkes@cbdiforum.com

¹ <http://en.wikipedia.org/wiki/Ecosystem>

² <http://www.acord.org/home.aspx>

³ <http://www.hl7.org/>

⁴ <http://www.sap.com/solutions/esa/index.epx>

⁵ eBay Developer Centre, <http://developer.ebay.com/index.html>

⁶ Amazon Web Services, <http://www.amazon.com/gp/browse.html/104-5044182-9217518?%5Fencoding=UTF8&node=3435361>

⁷ The use of ebXML and WS-STAR is further described in the CBDI Forum Report "E-Business Standards: The role of ebXML and Web Service Protocols"

⁸ <http://www.globus.org/ogsa/>

⁹ Java Technology and Web Services: <http://java.sun.com/webservices/index.jsp>

¹⁰ Microsoft .NET Framework Developer Center: <http://msdn.microsoft.com/netframework/>

¹¹ Apache Axis Web Services Project: <http://ws.apache.org/>

¹² Practical Service Specification and Design. CBDI Journal 2005.

http://www.cbdiforum.com/secure/interact/2005-03/Practical_Service_Spec.php