Workshop on Web of Services for Enterprise Computing

Fujitsu Submission v0.2

Authors:

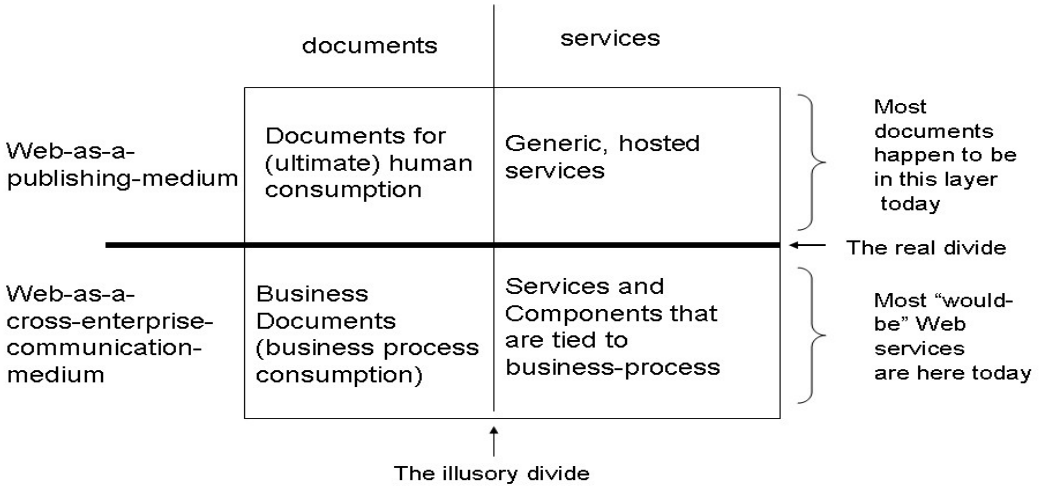
Jacques Durand
Tom Rutt
Hamid BenMalek

Acknowledgements:

Masahiko Narita
Paul A. Knapp

1. The Great Divide

The fundamental schism that has to be overcome for the Web to further enable enterprise business processes, is not so much between the Web of documents and the Web of Services, than it is between the Web-as-an-open-publishing-medium and the Web-as-a-cross-enterprise-communication-medium.



In the Web-as-an-open-publishing-medium:

- Both documents and services are resources directly exposed to the world, either for free or in a business model where more visibility, more users and easier access is better.
- The resources – document or service - being published are usually assets that have an intrinsic value : themselves are the goods being sold or offered and are not intermediates to further back-office operations of the publishing enterprise. - with the exception of the on-line stores (see below).
- The resource-providing company acts as a host, not a partner: E.g. Amazon mechanical turk or simple storage – two services invoked by applications - , are hosted services. Using these services even from a client business process is not motivated by cross-enterprise trading or business – it simply means outsourcing some function.
- Generic interfaces and protocols suffice.

In the Web-as-a-cross-enterprise-communication-medium:

- Both documents and services are resources that are only intended to a selected set of users and have their visibility and access controlled by contracts that may greatly between users.
- Not surprisingly, the documents and services have significant ties to the back-end business processes, which implies integration constraints, significant change management issues, on both ends.
- The resource-providing company is a business partner in the sense of the core business being conducted by the enterprise. The service or document has an effect on some core business process.
- Heterogeneity of needs and practices reigns. HTTP is not always the best protocol associated with this usage pattern. Bulk transfer of documents is sometimes necessary. Various levels of reliability, security or transfer control are needed.

In other words, the real barrier does not reside so much in the different natures of the resources - document or service -, than it resides in the often underplayed differences between two Web usage patterns. It just happens that the publishing usage pattern – the most successful and most understood as of today - overwhelmingly handles documents, as this is where the bulk of the demand is today.

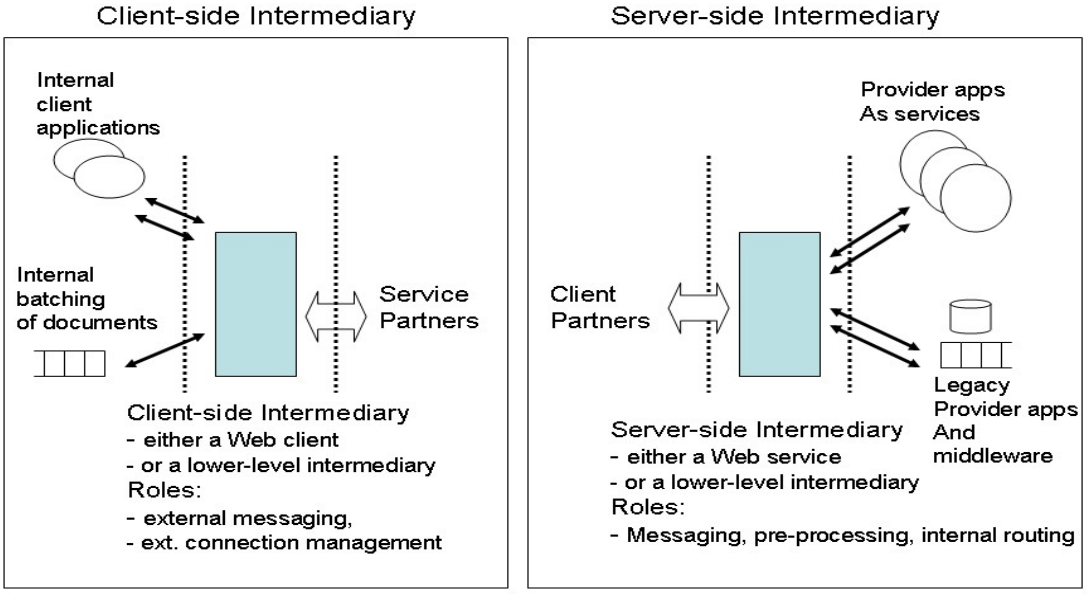
To be sure, some practices don't fall neatly in one or the other category. The on-line store model has ties in business processes but can be seen as an extension of the publishing pattern, adding an input channel. In the Software-as-a-Service model a service is hosted and sold per use but the outsourcing of vital business functions is more akin to cross-enterprise communication in a strong contractual context.

The point here, is that enabling the deployment of services for the publishing-medium Web usage pattern, and enabling services for the cross-enterprise usage pattern, are two different things. Similarly, business documents as consumed by business processes call for a different Web handling than documents for publishing.

2. Challenges from the Cross-Enterprise-Communication Web Usage Pattern

Consider the following use case from medical applications (HL7-related):

On server side: several health provider applications that may be specialized (Medical, Pharmacy, Dental, Information, etc.) or may just handle different "books of business" (Medical for company A, Medical for company B, etc.). These provider applications are deployed as internal servers. Requests to these providers are either document inputs or inquiries. They are mediated by a single front component that handles all external communications and related properties (security, quality of service, user management, initial validation, error detection and recovery exchanges) but also internal communication (content-based internal dispatching/routing, store-and-forward, message pre-processing). This front gateway acts as a server-side intermediary to health provider applications. Symmetric design may occur on client side, with a client-side intermediary.



The above server-side intermediary usually needs to operate as a Web service itself, in the absence of adequate intermediation technology at lower level (e.g. For content-based internal routing, initial validations, etc.). This model may cause all sorts of inefficiencies: in the handling of documents being forwarded, or due to the loss of header-level information, e.g. In case security credentials may need to be propagated beyond the intermediary, for fine-grained authorization at provider-level. In addition, managing change in this front service definition gets more complicated due to dependencies with many providers. Yet when the provider applications on the back-end are themselves wrapped as Web services – like in SOA environments -, the service-based intermediation either introduces redundancies or mismatches.

In short, the Web deployment of above enterprise services lacks support for the internal message process flow that takes place on either side. This is where cross-enterprise communication requirements meet back-end integration, and the "plumbing" technology on the back-end – SOA or other – has often been mistaken as the solution, while it is only the framework for it.

More generally, the challenges posed by cross-enterprise communication to the Web as a medium are:

- Transactions governed by contracts (as negotiated policies) affecting middleware functions (QoS...) in a way possibly tied to business content or to service specifics.
- Diversity of communication requirements (intermittent connectivity, bulk transfer, flow-control...)
- Legacy practices to deal with (in document representations, receipts, service response time and granularity) – no clean slate.
- Business-disruptive transitions and upgrades, high organizational impact.
- Robustness to upgrades needs change management, notification and multi-version support.
- Server-side as well as client-side message flows and intermediaries.
- Back-end integration constraints and variety
- Documents may have several levels of validation and authorization along the message flow.

Some lessons learned from use cases similar to the one above are:

(a) When binding a business document with a service interface, the later is often the better.

- In RPC-style service invocation, the XML schema associated with a request message for a service operation, is used in a type checking pattern by most Web service stacks: if the document inside the request does not match the declared schema, the request is simply rejected as an invalid message. While this sounds like a smart engineering principle, in practice this is often a horrible way to handle a business document. In many cases, schema violation should not be handled differently from semantic rule violation in a business document.
- When many complex, customizable document types are involved, the interface contracts associated with services (WSDL definitions) are a maintenance liability, creating more interoperability hurdles on the long run.

(b) Contracts are fragmented and too service-centric.

In the Web-as-cross-enterprise-communication the notion of contract – here governing interoperability and QoS - is key.

Support for contracts is however fragmented: the interface contract (WSDL) refers to document contracts (XML schema) that only cover a format – XML – and some syntactic and semantic rules, the rest of which is represented somewhere else. Other contract elements involve Policies. SOAP headers often express other forms of contract affecting QoS. This fragmentation becomes problematic in an intermediary model where different nodes do different validations, or may need to reuse or correlate contract elements.

(c) Message flow and pre-processing often require access to business data.

Internal routing, as well as quality of service may be dependent on payload content, E.g. The routing decision on server side for a Pharmacy form to the correct provider application, may be decided dynamically based on content. The privacy level of a Medical document may similarly vary with content. Validating compliance to these dependency rules is often necessary on the receiver side – e.g. associating requester credentials with the use of the right document or the right destination, is an entitlement issue that will affect routing.

3. Getting a REST or a Break?

REST is clearly the protocol of choice for the Web-as-publishing-medium pattern, while the situation is more complicated in the light of the cross-enterprise communication requirements. A look at the emerging third generation of Web services stacks gives valuable clues about what is in demand. In particular, consider Axis2 1.1:

- Either REST or SOAP may be used to connect to the same service, WSDL-based or not.
- AXIOM technology reduces greatly the parsing overhead of complex XML business documents, allows for partial or incremental processing, more immune to schema variations (versioning...). It also helps the selective extraction of headers during message flow.
- Services without WSDL. These document-centric services – with SWA supported in 1.1 - provide great binding flexibility on the back-end, and leave the communication contract to whatever agreement is attached to the document itself.

Similar evolution can be seen in other stacks. REST gets better integrated than in previous versions. With better support from SOAP 1.2 (and WSDL2) for REST style (e.g. HTTP GET), it is fair to say that future Web services are getting more REST.

And what about documents? To be sure, the concept of service has often been over-emphasized in the past to the detriment of document-centric processing. The new stacks appear to bring back more balance here. Whether using REST or SOAP, business documents are definitely getting a break here.

4. Conclusion

So doesn't all that speak in favor of REST? Not necessarily. While HTTP reigns in the Web-as-publishing-medium pattern, it is being challenged in some cross-enterprise cases. More importantly when considering the previous requirements, the future value of SOAP may be not so much about binding to a service interface, than in supporting message process flow and intermediaries - paramount in the cross-enterprise usage pattern of the Web. Although more remains to do on the intermediary model and composition of related functions, this is an area where the apparent simplicity of REST does not help much, while SOAP already laid the groundwork. How SOAP will fare in enabling message flow and service brokerage within coming SOA fabrics might be critical to confirm this advantage.

As for the Web-as-cross-enterprise-communication, two of the main challenges remain in our opinion:

- Better support for a robust "local" intermediary model supportive of message flow on each enterprise endpoint. SOAP provides the framework, current stacks the enabling mechanisms for this. (But where are the standards?)
- Better support for contracts, which means better integration of various elements (policies service contexts, rules..). Beyond a policy container and attachment mechanism (WS-Policy and related) more can be done in policy processing (negotiation, intersection,) before hitting the domain-specific barrier. A policy model (a la XACML) could help.