Services Orientated Architectures and Payment Systems

By Greg Brougham, March 2009

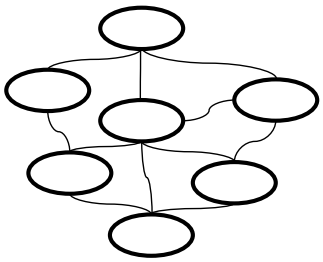
"*The real voyage of discovery consists not in seeking new landscape but in having new eyes*" A la recherche du temps perdu, Marcel Proust

Introduction

We are living in interesting and challenges times but the objective of an organisation remains to make money now and in the future. Payments are a necessity that supports the operation of the market place but for most banks payments, although critical, are not in general a profitable element of the organisation. Over the last few years SOA has become a buzz word and a number of vendors have promised that they would transform the payment estate resulting in significant benefit for most organisations. This paper reviews this proposition based on experience that we have gained as an industry. It should be noted that most technologies take about ten years to reach maturity – the first release of Java was available in 1992 but it was not until around 2002 that it started to become the de facto standard for enterprise developments. Similarly version 1.0 of the XML standard was released in early 1998 and as a technology it had it 10 year anniversary just last year [1].

Context

Payment systems do not exist in isolation of the organisation which makes use of them. If we understand the organisational structure of an organisation then maybe we can understand how the payment systems should be structured and thereby have a clearer understanding of the benefits of SOA in the payments arena. Tappin and Cave [2] use the cell metaphor to describe an organisation. They go on to talk about a non-hierarchical management structure that they believe is necessary given the challenges facing today's enterprises. Bennis [3] draws a similar comparison and believes that organisations need to be more organic. This is not new as systems thinking has long held the view that every living thing is connected by relationships.



Payments systems support these business enterprises and we have long considered these systems to be of a systemic nature. Goldratt [4] believes that such systems, although they are complex are also simple. This is because all elements of the systems are interconnected by cause-effect relationships and therefore the system has only a single degree of freedom. So, although we can decompose payments systems into a series of services to support individual business functions we need to acknowledge the complexity and need for these to reflect the structure and boundaries of the organisation. Organisations are composed of a number of elements that are connected via relationships and this provides the context for their existence [5].

SOA has been pitched as the saviour of the software industry in general and some vendors Sun and IBM have developed specific frameworks[1] for the payments industry. The approach is interface

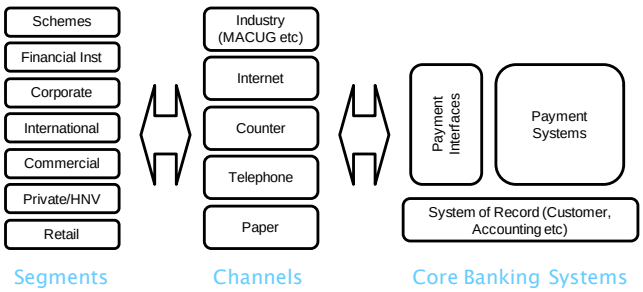
[1] IBM's Enterprise Payments Platform and Sun's Open Payment Bus

focused and based on loose coupling of distributed components that are referred to as services. These services can be realised with different technologies but the protocols for the exchange of information have been standardised and typically use web standards. The approach is vendor and platform agnostic and supports the majority of existing and current systems. It doesn't matter if the platform only supports COBOL and is 30 years old as the same solution can be provided and all these systems can be inter-connected using these standards. They support a degree of self-organisation (can be orchestrated), they exhibit interdependence (support composition) as more than one service is required to realise a business function and they are by nature diverse (alignment) as they can be offered by any number of business units to support any number of business functions and capabilities. This means that they reflect the connectedness, relationships and context that are the representative of an organisation [5].

The fundamental principle is not dissimilar to what enterprise architects have been expounding for a number of years – namely that the system should offer functionality that supports the capabilities of the business units. Microsoft found this when it first began talking to organisations about its Motion framework [6] as it demonstrated the value of such an architectural approach. What the framework did was to provide a decomposition of the enterprise as a set of cooperating capabilities grouped by business function. These capabilities can be realised as services as part of a service orientated architecture. It should also be noted that it is not necessary to have the complete picture but only to have defined the boundaries for the particular business unit that needs to be supported[2].

Payment Systems and SOA

We have already noted that payment systems by their very nature are systemic in nature and for the majority of banks they are core[3]. These systems have evolved over a number of years and therefore entail a large number of disparate systems, involving many platform technologies[4] and are composed of legacy in house developments and third party commercial applications.



SOA has been sold as having significant benefits for any organisation – increased ROI, lower maintenance, higher productivity, more re-use, improved system/business alignment, increase agility and exploitation of existing infrastructure to name a few. These benefits are largely achieved through the use of standards to support looser coupling of the systems in a vendor agnostic manner.

The industry pitch is, that as the concepts are vendor and technology agnostic they can be applied to any computer systems. We should expect to see benefits specifically in a number of areas such as

[2] This avoids the catch 22 situation of needing to have defined the enterprise architecture before exploiting the technology
[3] In the case that the bank is a clearing bank
[4] Typically including Tandem, mainframe, Unix and Windows

improved re-use, high productivity, reduced time to market, increased ROI, lower maintenance and increased flexibility.

Improved Reuse and Higher Productivity

This comes from the alignment of the business domains/capabilities/sub-units and the architecture of the individual systems that support them. If the services are well defined and aligned to the business then they can be used by multiple elements of the organisation. The business will change with time but the core capabilities tend not to change over time which leads to improved reuse. Importantly alignment can be used to scope and bound work, ensuring that inappropriate business functions are not outsourced!

Time to Market & Return on Investment

The services offer certain functionality or support individual capability so the granularity of change is smaller. This is expected to lead to shorter time to market as the scope of individual changes are smaller. In addition these services can be orchestrated to provide new functionality without the need to write new functionality. This is analogous to the mash ups that are appearing on the internet where new capabilities are offered without changes to the core services of existing systems.

If we understand the boundaries of the business capabilities it is not necessary to implement these capabilities within the organisation. The cost and strategic alignment can be assessed and if it is considered appropriate the capability can be outsourced or offshored.

Lower Maintenance and Increased Flexibility

The increasingly granular nature of the resultant systems results in more flexibility and lower maintenance. Instead of changes to monolithic applications, the services can be changed individually. Testing is simplified as only the individual and dependent services need to be tested.

Increase in flexibility comes from the component nature of systems. The components or services can be used to support new business functionality overtime as the changes are smaller and more granular in nature.

The Reality

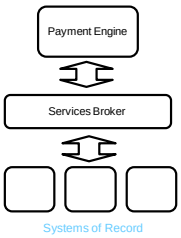
So have we realised the benefits? – largely no. The nature of the payments estate and the complexity of change has largely compromised the value realisation for the majority of large banks and has led to little benefit. This has resulted largely from the effort and time required to re-engineer the existing monolithic applications that compromise the majority of the payments estate. There are four particular areas that appear to have led to this situation – namely integration, governance, delivery approach and technology.

Integration

Payment systems are typically at the core of most banks, as noted before, and therefore the payments systems typically integrate with the majority of the channel and product systems. The boundaries of these systems, although defined, use proprietary interfaces and standards. This makes

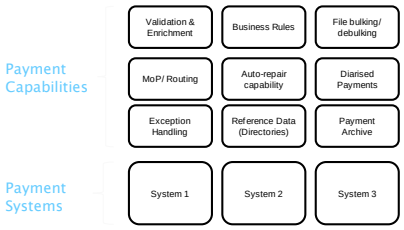
changes to systems complex and time consuming due to the testing requirements. To start to adopt an SOA based approach, a set of new services have to be defined that provide generic services that support the connections to the channel and product systems. These services need to be coarse grained and reflect the capability boundaries of the business units.

The value of composition or service orchestration is only available when there are a set of suitable services that can be exploited. As we are still effectively in the build out phase we still tend to see these used at a one to one level to support existing business capabilities that are being re-engineered. Hierarchies tend to be simple and limited for the moment.



Governance

What is obvious is that governance needs to be pragmatic as this is a paradigm shift. Clive Longbottom [7] has drawn attention to this in an article on The Register in which he compares the Scandinavian and UK experiences. The penetration of SOA in the UK is quite low while it is much higher in Scandinavia. The key point, is that in the UK, the view seems to be that you need have defined the enterprise architecture before you can deploy SOA. The Scandinavian's in comparison have been much more pragmatic and have realised that you don't need to have defined a model for the whole enterprise but only for the business area for which you wish to start exploiting SOA. This allows an organisation to start experimenting and learning without the overhead and cost of a full governance model. This is much more pragmatic and I expect to see more organisations following this approach.



A related area to this is the area of governance related to the services contracts - central registrar or distributed. Here I side with Webber [9] in that I believe it should be the responsibility of the domain/business unit to maintain the services contracts. When it comes to external relationships then a centralised registrar makes sense as we may not know who owns the services that we are consuming. For internal systems we have defined owners and they will know which services, and which version of these, are applicable. Putting this in the hands of a third party is only adding overhead and reducing the flexibility of the organisation.

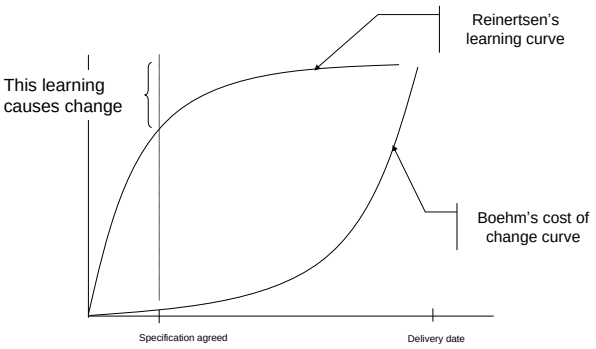
Delivery Approach

The majority of organisations are typically still using a SDLC methodology for system delivery. This is at odds with SOA and compromises the time to market, ROI and improved value delivery. Some of the benefits, such as flexibility and improved maintainability may be realised in the medium to longer term but the business value in the short term is compromised.

A key point is that the services amount to contracts and therefore the process of definition can be onerous[5]. Since we are typically using WSDL to define these services this means that the contract has to be in place early in the design phase to support the building of the service. This defines the interface early in the development cycle and therefore reduces our options. The implications are

[5] Thomas Erl and et al.'s tomb *Web Service Contract Design and Versioning for SOA* runs to 800 odd pages

that we may have to change the interface as we learn more about the service (right left validation of the contract) and therefore delivery may be impacted[6].



A comparison of Boehm's cost of change [9] and Reinertsen's information arrival [10] curves draw out this paradox. Do we fix the specification early when we run a risk that it will be wrong or do we defer it till later? We make good decisions when we have experience, are well informed and have prompt feedback [11]. We need to be more flexible as perfection is the enemy of good enough. If we can break down the delivery steps into smaller chunks based on what we know we can improve the value realisation. Elements of the system are delivered earlier and we have the opportunity to learn from the experience and improve the quality of the system. This is what iterative and incremental delivery approaches support.

Technical Standards

The technology standards in the area of Web Services have taken time to mature. It has also taken time for the software development vendors to integrate the technology into their tooling to support mass market adoption. These delays have limited the productivity improvements and has lead to interoperability issues. Most web services have now moved beyond the 1.0 release.

We all know that XML is flexible and has recently had its 10th year anniversary. In terms of technology it is reasonably mature as we have already mentioned but there are still challenges. One of these is in the area of performance and scalability. We have moved a long way from the use of BACS type 18 fixed field length message to dynamic and variable length fields. These are significantly more expensive to process and this has implications in terms of scalability of the systems that process the messages and for the scheme themselves. This is an area where we may have to rely on hardware accelerators to ensure scalability. Canonical message structures have a lot going for them!

You also don't need the latest greatest ESB technology to support the use and deployment of SOA. The key value comes from adoption primarily of the principles of exploitation and delivery based on services. The technology can be what the enterprise can support. Jim Webber draws a comparison between the old point to point connection of old, versus the vendors ESB products – the only difference is that the spaghetti is now contained within the vendor's proprietary technology and is no longer visible. You have effectively outsourced the management of this to the vendor if you go down this route. The alternative is to accept this and consider a simpler model along the lines of the REST model. You will need to consider this as part of your technology strategy.

[6] Alternatively we can version the service but we really what to control this as we don't want a proliferation of definitions in the running system that are essentially the same (probably no more than 3 in practice).

So What Can We Do?

So before concluding what are the key challenges that we face and is there some value to be taken from the experience to date? With regard to the service definition we can adopt a standard approach. The process can be kick started by using an industry or proprietary framework such as IBM's IFW or Microsoft's Motion Framework but note that these typically have to be extended to support the internal functions of an organisation. If you are not in a position to leverage one of the vendor frameworks then there is always Eric Evan's Domain Driven Design practice which has been used at a strategic level [11] to support the capabilities definition.

We also need to ensure that the systems are stateful and store the transaction. There is the example of Sainsbury and Tesco's online shopping experience in the UK. In the case of Sainsbury the early application held the shopping basket but it did not retain state so when the user connection was lost all the contents of the basket was lost. This was in the early days of online shopping in the UK and it was not untypical for the connection to be lost. What the application really needed to be is idempotent. This means that it should be able to detect and handle duplicates in a sensible manner. This is analogous to how a business works. If a business was to receive an invoice it would validate it and ensure that it was not a duplicate of the one already processed. This resilience at the functional level makes the system more flexible and resilient overall.

Although the core web services standards are reasonably mature the areas relating to security are still developing. Again this is not a major issue for the majority of companies as they tend to be quite mature in terms of application and network security. The existing solutions can be applied to the internal systems and for external systems SWIFT can used as the transport. This implies that we are okay when the network is closed. This is likely to remain an issue for sometime as payments systems are typically core to the bank and therefore banks are unlikely to want to connect these critical systems to the Internet. We can of course exploit MA-CUGs and SCORE for such a purpose but ultimately we are going to have to expose some services related to payments on the Internet. This will be an area to watch closely and one for competitive advantage.

Other areas that are still in need of development are related to the semantics of services interactions and versioning. The existing standards do not really address the conversational aspect of the exchange of information. These aspects relate to the navigation of services that are required to undertake a business function such as "pay an invoice". I can submit the invoice but the service definitions don't necessarily tell me what related services and in which order they need to be called to complete the processing of the invoice and to check whether it has been paid. For web applications this is implicit in the conversation as buttons (links) are presented on the page and buttons for functions that cannot be executed would typically be greyed out. Related to this point are the actual error codes and the semantics of recovery. There is a lot of information that needs to be provided in order that the services can be used effectively. There are standards that are offered as part of WS-* but these are only supported by the framework tools such as vendor ESB offering. Another area that has not really been address is the area of versioning. At the moment this is left largely to the individual implementer to support as there are no standards just approaches. We have a long way to go before we reach the level of maturity of older technologies such as CICS and DCE.

The last area that is still to be fully addressed is scalability of applications due to the overheads related to XML. The message structure is flexible and but as pointed out earlier this is a long way from BACS type 18 messages. They are expensive in terms of processor cycles to process. We can address this by offloading the processing to some external hardware devices and these have moved into the mainstream over the last couple of year. These devices can transform XML messages into some canonical data structure that is more closely aligned with the target systems and therefore reduce the overall latency and improve the system scalability.

Conclusions

It is not surprising given the systemic nature of payments and the maturity of the technology that SOA to date has not had a significant impact on the industry but we are reaching a tipping point. Gartner coined the phrase and described Service Orientated Architectures in 1996 [10], that is now over a decade ago. The standards are achieving a level of maturity such that the use of the technology within an organisation is viable and the architectural principles that underpin SOA are sound and have value. A number of issue have compromised the value of SOA to payments, primarily -

- Payments are at the core of most banks and this has lead to issues due to the scale and complexity of the changes involved
- The governance model needs to be pragmatic and it is not necessary to have the full enterprise architecture defined before SOA is exploited
- The traditional delivery approaches focus on requirements specification early in the delivery cycle which compromises the value of SOA. The use of iterative and incremental delivery approaches needs to be exploited to gain better value realisation
- The maturity of the technology has been overstated but most of the vendors now support the majority of Web Service standard and these standard are reaching an usable level of maturity

Payments systems are systemic in nature and change has to be justified in the context of the business. Initiatives such as the formation of SEPA don't happen very often but the scale of the impact on bank systems in general mean that re-architecturing as part of the changes to meet the regulatory requirements is unlikely to be an option. The adoption of SOA is more likely to happen when there is a major refresh or renewal of the existing systems.

There are benefits be achieved from SOA, namely in terms of flexibility and improved alignment with business capability and the life span of systems. These benefits will only be tangible when a significant amount of the existing estate has been modernised and will only be realised in the future when the systems needs to evolve to meet changing business requirements. The continued use of traditional SDLC delivery methodologies has restricted the benefits of SOA, such as reduced time to market and increased ROI, and lead to some degree of disillusionment.

None of these are reasons not to adopt SOA, just realise that the benefits may be longer in coming if you cannot exploit a pragmatic approach in terms of integration, governance, delivery approach and technology.

References

[1] Bray recalls team XML, should a happened sooner. Phil Manchester, Feb 2008 - http://www.theregister.co.uk/2008/02/15/xml_tenth_anniversary/

[2] The Secrets of CEOs. Steve Tappin and Andrew Cave, 1998

[3] Why Leaders Can't Lead. The Unconscious Conspiracy Continues. Warren Bennis, 1997

[4] The Choice. Goldratt, 2008

[5] Profit Beyond Measure, Extraordinary Results through Attention to Work and People. H. Thomas Johnson and Anders Broms.

[6] Microsoft Motion Business Modelling Methodology. Microsoft, Ric Merrifield, 2006

[7] SOA dead or alive? Can it deliver on its promises. Clive Longbottom, Quocirca, June 2007 - http://www.theregister.co.uk/2007/06/14/soa_comment/

[8] Service Contract Design and Versioning for SOA. Thomas Erl et al., 2008

[9] Software Engineering Economics, Barry Boehm, 1981

[10] Managing the Design Factory, A Product Developer's Toolkit. Donald G. Reinsertsen

[11] Nudge , Improving Decisions About Health, Wealth, and Happiness. Richard H. Thaler and Cass R. Sunstein, 2008

[12] Developing Enterprise Web Services, An Architect's Guide. Sandeep Chatterjee and James Webber

[13] Architectural Improvement by use of Strategic Level Domain-Driven Design. Einar Landre, Statoil ASA, Harald Wesenberg, Statoil ASA and Harald Ronneberg, Statoil ASA. See http://domaindrivendesign.org/practitioner_reports/landre_einar_2006_part1.html

[14] Service-Oriented Architecture Scenario, Gartner. http://www.gartner.com/DisplayDocument?doc_cd=114358. The original research articles are SSA Research Note SPA-401-068, 12 April 1996, "'Service Oriented' Architectures, Part 1" and SSA Research Note SPA-401-069, 12 April 1996, "'Service Oriented' Architectures, Part 2" - (provided for historic perspective)

Revision History

Version 0.3, March 2009

About Greg Brougham

Greg Brougham is an experienced systems architect with extensive experience of payment systems. He has over 20 years of experience in the development and delivery of such systems for large financial institutions and market infrastructure companies. This included two of the largest payments developments in Europe in the last decade, namely the initial delivery of the NewBACS clearing and settlement system and the development of the new card clearing and settlement system for Visa Europe.

He previously worked for Accenture and was responsible for the technology strategy for one of the UK operating units. He is interested in system architecture, in particular the areas of high availability, scalability and system longevity. Greg is currently involved in the delivery of a high-care payment system, based on FUNDtech's Global PayPlus, for one of the major UK retail banks.

Email@greg.brougham@tadlon.com