

Pragmatic SOA in the payments industry

by David Chance

"A complex system that works is invariably found to have evolved from a simple system that works."

John Gaule (1603-1687)



Dovetail
Progressive renovation in payments.

info@dovetailsystems.com
www.dovetailsystems.com

Europe

Dovetail Systems Ltd.
101 Morgate
London, EC2M 6SL
UK
T: +44 (0) 20 7997 1200
F: +44 (0) 20 7826 4440

North America

Dovetail Systems Inc.
130 Clinton Road
Fairfield, NJ 07004
USA
T: 1 973 882 9922
F: 1 973 882 9822

The SOA approach to IT design and development reaches out from traditional IT silos to business groups, to define the services they require. SOA promises to improve an organisation's IT efficiency, and subsequently its business agility by targeting reusable services. It can help drive the deconstruction of operating silos, improve IT and business alignment, speed development of new products and services, and allow banks to bring new products to market more quickly. This paper focuses on our experience of bringing SOA principles to the implementation of large scale, global payments systems, and on the design choices we made in doing so successfully.

With the drive towards reusable, service-based software coming from within IT, SOA is often associated with a particular set of technologies or tools (e.g. WebServices, BPEL, BMS). But SOA is actually an approach to defining how business processes within an enterprise interact, how existing assets can be re-used to reduce risk and how these business services can be shared to reduce costs and increase business agility. The tools behind SOA are enablers – they should not be the primary drivers for adoption.

It is important to understand that, although technology is key to SOA adoption, it has to be balanced against the specific business needs of the organisation. This allows a pragmatic approach to be taken, not just in choosing the right technologies for a specific task, but also in deciding where and when to adopt SOA principles to maximise business benefits.

Analysing a business to dissect the business processes at a level that has meaning for both the IT and business is critical for success. The tools used should be chosen carefully to ensure they meet the demands of the particular business domain and to enable the level of abstraction required.

A brief history of SOA

Service Orientated Architecture is the result of over fifty years of evolution in the way systems are designed and software developed. Crucially, it embraces the hybrid nature of modern technology infrastructures, and provides a model in which ongoing, stepwise improvements can be made. The driving goal is reuse: the ability to build modules and components once and use them many times across the enterprise to increase business agility, at the same time

reducing risk and IT costs. It also encourages wrapping existing, legacy functionality with service interfaces so that it can be leveraged in migrating an overall infrastructure step by step.

The evolution of SOA started in the 1950's when the IBM SHARE user group was formed to share binaries between members; followed in the 1960's by Universal Data Structures that provided shared data schemas, which – for the first time – allowed applications to share logic. The next major step was in the 1980's with the development of shared application libraries (C library) and the principles of context-less module development controlled via transaction managers (e.g. Tandem NonStop Pathway). Thanks to Moore's law (that computing power will expand exponentially, doubling every 24 months), the 1990s saw immense growth in the computing power available to organisations, at rapidly reducing costs. The cheap server-based computing platforms led to the development of distributed, component-based systems. The ideals behind Component Based Development (CBD) led to the technologies that underpin SOA today.

What are the advantages?

SOA promises to deliver major benefits when used in the payments industry and adopted by both the business and by IT. The main benefits:

- Business agility – the ability to quickly modify existing applications, and create new composite applications when market trends and competition demand change
- Business-focused IT development – the organisation of services around the way the business operates, providing a level of abstraction that helps bridge the gap between business understanding and IT implementation
- Reduced risk and costs – the re-use of proven, existing assets reduces redundancy and aids rationalization
- Location independence and distributed processing – the capability to scale and allow operations to be sited at the most effective locations
- Standardised interfaces – the streamlining of integration between systems, reducing testing and compliance overhead and allowing renovation of existing systems in a controlled way

Creating functional components that have the right, business-meaningful abstractions to support the business domain allows business agility and significant reuse to be achieved. Where there is sufficient external standardisation, this can be achieved using external components and multi-vendor solutions. A good example of standardisation and simplicity of interface that allows the use of components from multiple vendors can be found in the audio industry. In the past, radiograms or hi-fis were produced that contained everything needed to play music from a variety of sources (e.g. radio, vinyl). A level of functional abstraction has taken place, allowing you to buy hi-fi components that perform a single function (e.g. mp3 and blu-ray players, amplifiers, wireless receivers, etc.) with a simple interface allowing these components to be bought from a range of suppliers with the knowledge that they will work together. External orchestration is provided by universal remote controls, and location independence is provided by the move toward networked and Wifi music systems.

Why SOA for payments processing?

SOA principles provide a natural fit with the needs of payments processing. Although the traditional approach has been to build large, discrete payment solutions for each payment product or clearing channel, payment processing systems are nevertheless similar in overall functionality and underlying requirements and, thus, contain a lot of redundancy.

Within a bank's payments infrastructure there is significant opportunity for reuse. The various business processes and system functions required to process different types of payment can be built from a set of common processes that differ only in the detail of their business rules (such as validation and qualification rules). Hence there is the potential to share well designed common processes across the various product or payment types.

On a wider scale, although built and implemented as discrete solutions in product silos, payment systems have to interact with numerous other systems within a bank to ensure the business integrity of the payment. Enterprise-wide systems, such as accounting, risk management, fraud control, funds management and foreign exchange dealing, as well as channel-specific delivery systems, are all required to process a single payment.

SOA principles are as applicable in sharing components within a core payments application as they are in orchestrating the interaction between shared services at the enterprise level. The technology and toolsets used, however, may be quite different.

There is a core set of capabilities that should be seen as fundamental for modern, efficient and profitable payments systems. These include:

- Business agility and separation – the ability to quickly respond to market changes and take advantage of market opportunities without affecting existing payment processing
- Straight Through Processing – minimising manual intervention to reduce the cost of processing. STP can be improved by validation on input, automated enrichment and repair, all of which minimise exception processing
- Visibility – enable business operations managers to have visibility, transparency and full information of the status and performance of payments
- Maintainability – regulation and compliance changes require payment systems to be regularly updated. Updating and testing systems can make up a large part of the operational costs
- Performance – payments systems measure performance both in terms of speed (with online, real-time payments with sub second end-to-end times) and volume (being able to process very large bulk payment files at the same time as individual messages without impacting system performance and meet customer service level agreements)

Of these, it is performance that is critical in the choice of appropriate tools.

Banks have to provide these core capabilities and, at the same time, cope with the overriding business necessity of containing costs and exceeding customer expectations.

SOA underlying technology decisions

There is a wide range of technology and toolsets with which to implement SOA. For some, the selection of the best technology and toolset is a matter of doctrine. This can be dangerous. To a large extent, the range of tools reflects the wide range of application types to which general SOA principles can be effectively applied – a large enterprise HR system has very different characteristics to a low latency trading system, or to a transaction-oriented payments processing system.

When deciding on a toolset it is important to take a pragmatic approach – decide what the goals of the organisation should be, the shape of the final solution and what constitutes the success criteria for the implementation. Following a dogmatic approach to choosing a toolset is unlikely to be successful, with arbitrary decisions being made to bend the requirements to meet the limitations of the tool. The pragmatic approach is to use the tools that best fit. Consideration should be given to:

- Service granularity
- Distributed processing
- Service interaction
- Interface standardisation
- Business agility
- Performance and throughput

Service granularity

At the core of an SOA infrastructure is the creation of abstractions that define the business processes to be encapsulated or developed as loosely-coupled services that can be reused and shared across multiple business solutions. The choice of granularity should be driven by identifying the key abstractions in the business domain. Services are often described as either being coarse or fine grained. This is a misnomer as the granularity of services should match the level of abstraction required for the business domain.

Ideally, from a logical perspective, a service should be fine grained enough to encapsulate specific business functions allowing the business to change the precise function they need rather than having to replace a significant percentage of unrelated functionality (just because it happened to

be in the same bucket), but not so fine grained that the complexity of assembly and orchestration becomes a management issue in itself.

The goal is to enable 'simple' business changes to be simply made, and make the sophistication available to make more complex changes when required.

The overwhelming caveat when determining the level of abstraction for a business process is performance. We will discuss this shortly.

Distributed processing

In a distributed environment, different functions can be implemented on different systems. The mistake often inadvertently laid out in the architectural vision of a future state is to conflate ideas of logical and physical separation. Physical distribution has to be carefully considered. While it may make sense for technology or operational reasons, there is a cost – specifically a communications cost.

The communications overhead in distributing complex and computationally intensive particle physics calculations based on data from CERN is relatively low, making such an endeavour viable. Orchestrating individual processing stages for a single payment is quite another matter.

Care should be taken when deciding what level of separation makes sense logically and physically. Logical separation should focus on defining services to optimise business manageability and agility of the services. Physical separation may be required when different technologies and systems interact; and may be highly desirable to enable horizontal scalability, and operational independence (key when business drivers and roll-out schedules may differ).

Service interactions

For a system built from shared services to function, the services must be able to interact in a coordinated way. This requires inter-service communication and orchestration. Communication can be either synchronous or asynchronous.

- Synchronous calls are simple to program and efficient (assuming short call times and smooth flow without processing backlogs)
- Asynchronous calls are more complex to program but give better visibility and control of flow and more opportunity to optimise block operations.

There are many methods that can be used to enable these interactions, including message orientated middleware, file transfer, and, local and remote procedure calls. It is important that the right method should be chosen for each situation within the overall infrastructure. For example:

- Asynchronous interaction, using messages and files, may be appropriate for payment flows and interaction with other systems, as it allows flow control to be monitored and managed
- Synchronous interaction, using EJB and Web Services, may be appropriate for external systems interacting with a service encapsulated within solution

Orchestration can be through generic SOA technology using standards such as BPEL, or through more specific, lightweight tools.

Standardised interfaces

Without a standard interface it becomes very difficult to share and reuse a service. Anything that wants to use that service has to have full knowledge of the interface and be updated whenever that interface changes.

Business agility is gained through the ability to share and reuse components and services and by having standard canonical interfaces to ensure this.

It is important that these interfaces reflect the needs of the business domain and are not made to be too generic across the enterprise. Focused, business domain aware, canonical interfaces allow the level of abstraction and interaction required for performance and manageability, whilst supporting the reuse and sharing required for agility.

Complying with external standards (such as ISO and SWIFT) where appropriate improves wider interaction with external systems, allowing the organisation to interact with external parties using a standard and accepted interface.

Business agility

Increasingly key to the success of a payments business is the ability to react quickly to changing trends, market opportunities and regulatory intervention. Ensuring that the abstractions chosen are the right shape and level of granularity to allow services to be reused, adapted (by making precise, surgical changes), augmented, or replaced, enhances the agility of the business to adapt to change.

Finer grained services allow more focused changes to be made with tightly defined testing – reducing the impact that forced changes, such as regulatory compliance, have on the system.

To deliver business agility, a framework must be appropriate for the domain in which it is used. It should hold rules by which the business operates in a way that can be managed and controlled. In the past, business rules were sometimes abdicated to a generalised rules engine, but it was quickly identified that this led to a maze of unmanageable rules that were difficult to change and test, resulting in a loss of agility.

Choosing the right abstractions also provides an effective language for communication between the business and IT. This common understanding allows a more efficient and innovative relationship to develop. The collaborative approach enables faster response to business change, and efficient development of new systems and replacement or reuse of existing ones.

Performance and throughput

Balancing performance and throughput against flexibility is vital to providing cost-effective capacity and reliable client service. Performance and throughput can be affected by many factors, including the level of business abstraction, the technology and toolsets used, the required resiliency and the actual distribution of components.

Performance is the most significant constraining factor on the choice of tools. There needs to be an awareness of the performance overhead when breaking down functionality within the tool set chosen. And there is a need to choose that toolset so that these performance constraints do not get too much in the way.

Some toolsets provide the required level of flexibility at the expense of performance. These add a level of overhead that makes it costly to provide additional throughput to cope with peak volumes.

In addition, SOA based systems are still constrained by the same performance barriers as traditional applications:

- Locality of the components – services that need to communicate across physical barriers will carry an additional overhead. Understanding that logical

distribution does not have to relate directly to physical distribution allows services to be located in proximity to each other

- Databases – database updates and commits are the most common performance-limiting chokepoints in transaction processing applications. Designing systems that can process multiple business transactions within a single database transaction minimises this problem

Why hasn't there been universal adoption?

The traditional approach within the payment processing industry has been to develop large, discrete payment systems for each payment product within each product or market segment, to a large extent reflecting the organisational structure in which the systems were built. These payment silos are limited in their awareness of the impact of change beyond their own operations. It is only recently with the focus on rationalisation, cost reduction and the success that some banks have had when adopting a universal payments engine, that management time has become focused on removing the artificial barriers between the different payment groups. It takes time to overcome internal politics and gain a common understanding on how to proceed.

Banks have invested heavily in their existing infrastructure and need to see a return from that investment. Wholesale replacement isn't usually an option – not just because of the costs involved, but also the associated risks. The continual imposition of governmental regulations and market compliance requires continual investment. It is usually seen as lower risk to make the investment in the existing system (one that the bank's own staff is familiar and comfortable with) rather than investing in a new solution. Such continual tactical activity keeps IT departments busy without achieving any strategic progress. Taking a progressive approach to renovation, however, allows a bank to invest in solutions that alleviate the pain whilst laying the foundations for future changes; in effect, using tactical projects to tack toward their strategic future state.

Putting SOA into practise – what has Dovetail learnt?

The Dovetail Payments System was designed and built from the outset with the goal of bringing the advantages of SOA to high-throughput, online transaction processing, and in particular to payments. It has been used successfully

to implement core transaction processing in payments in some of the world's largest banks. This experience has shown that a pragmatic adoption of SOA principles can work effectively, both for the bank and for the supplier in a number of areas: delivering business agility; separation and encapsulation of local change; straight-through processing; transparency of operations; maintainability; and, not least, the ability to scale performance as required.

In making design and technology choices in the Dovetail Payments System, we put considerable effort into defining the right business-meaningful abstractions, and in selecting the most appropriate tools. We have found it critical not to be tied down by the dogma of a particular toolset or technology, but to use what is most appropriate in any given situation.

Some of the key factors in making this successful include:

- Within the application, we forewent technology portability for all inter-service communication in favour of a component model that delivered the necessary performance by communicating through object passing when components were co-located. We used a lightweight workflow engine to provide orchestration and transaction management, enabling us to improve business agility without sacrificing performance. Communication with external services within our workflows was managed through Web Services, EJBs or, most commonly, message orientated middleware
- Building a payments specific framework. We defined key abstractions that enabled re-use, including:
 - a canonical payment object that was applicable across all high and low value payment types (i.e. the complete business context around processing a payment, including, but not limited to the external messages related to it)
 - processing steps in workflows (e.g. funds control checking, ATR matching, and conditional hold), gateways and long-running services (e.g. liquidity manager, and bank codes directories)
 - local rule sets for specific variations in processing (e.g. in qualification, validation, verification, intervention filtering, routing and re-routing)
 - product and client specific preferences that vary the standard processing

- Designing from the outset with performance and scalability in mind. This resulted in a highly scalable architecture with the following characteristics:
 - Data only serialised when absolutely necessary, only stored when required for transaction integrity and recoverability
 - Main message and file communication handled asynchronously so that queue management (even within components of the application) is visible and can be managed and controlled in operation. This also allowed significant optimization of flow to be applied at the framework level in both communication and storage
 - Storage scheme optimised for performance, with sophisticated distributed caching so that payment objects are rarely read from the database during normal operation
 - Application logic run in stateless compute nodes with configurable thread pools to scale vertically and take full advantage of hardware, and support for clustering to scale horizontally
- Designing from the outset with resilience in mind. This resulted in a simple state model with well defined capabilities for High Availability and Disaster recovery. It also enabled banks to use a variety of standard tools for managing their preferred resiliency model
- Using the most appropriate technology and toolsets in each situation. This resulted in a hybrid technology stack to deliver the optimal balance between business agility and performance:
 - Within the application, components are orchestrated using a lightweight workflow engine and data is passed as objects in memory. In this way, the granularity of application components could be chosen to best fit the business with almost no impact on performance, and operations could get a fine grained view of data flows within the application
 - At the enterprise level, between our framework and other systems, we used heavier weight technology such as Web Services to provide open access to services (real-time business data and logic)
- Automated testing was seen as a necessary part of the overall solution and not just a 'nice to have'. Without it true business agility is compromised by time-consuming

manual regression testing. This resulted in great care being taken in the structure of business requirements and how they mapped into testable conditions. We also built a sophisticated test environment which, like the framework itself, was rich in payments specific capability. This has proved invaluable in translating testable business conditions directly into automated test scripts that are readable by business analysts. The environment is now in use for continuous regression testing by us and onsite by our clients

Conclusion

Pragmatic adoption of the principles of SOA is viable for renovating even the largest, highest throughput payments environments. Moreover, we believe that such an approach is key to delivering the advantages banks require to compete effectively in the payments market. Pragmatic SOA leads to a better and more flexible implementation of technology that improves the agility of the business, whilst reducing both risk and operational costs.

But unthinking adoption of SOA technology and toolsets results in too many compromises to the business architecture (e.g. resorting to too coarse grained abstractions in order to achieve some level of performance) for the true benefits of an SOA approach to be attained.

Key to a successful SOA approach are:

- Identifying the right abstractions – have a business-aware framework that breaks down into components/ services that are both business meaningful and technically practical
- Using the right tools – for technical integration, performance etc, and understand that the tools being used can be different at different levels (within the payments application, and enterprise wide)
- Setting an achievable scope – take steps that can be completed and validated in relatively short time frames

Many vendors talk about SOA and have implemented wrappers for their existing legacy applications using SOA technologies such as Web Services. This can help with integration of these legacy systems into an enterprise wide SOA infrastructure and support migration from them onto more modern systems. It does not however provide the benefits of SOA for the business functionality encapsulated within the legacy system.

Our experience with pragmatic SOA and progressive renovation has shown that, as integration with bank systems progresses and bank-specific processing extensions to the core business services are completed, the pace of functional change accelerates. This is the direct opposite of the traditional world where increasing levels of change and complexity tend to scar applications over time, reducing their flexibility, and thereby decreasing business agility, lowering productivity and increasing costs. Sadly, a survey of bank payments systems today would provide more evidence for this last point than for the benefits of SOA. We firmly believe that this will have to change for banks to remain competitive.