2008 IEEE Congress on Services 2008 - Part I

The Role of Service Granularity in A Successful SOA Realization – A Case Study

Naveen Kulkarni, Vishal Dwivedi
SETLabs, Infosys Technologies Ltd
{ Naveen_Kulkarni, Vishal_Dwivedi}@infosys.com

Abstract

This paper presents the case study of a leading US financial institution International Financial and Brokerage Services (IFBS),[*] *which faced SOA realization issues while it followed an in-appropriate SOA design strategy. While riding on the SOA hype wave, it implemented thousands of fine grained web-services without paying much heed to issues like governance, and usage within its business processes. IFBS's service portfolio comprises of a gamut of services which although on paper looked good, but presented a lot of challenges in their usage and maintenance. We address some of these issues in this paper and present our framework for SOA adoption (called INSOAP) which can be effective in a similar end- to-end SOA adoption exercise in an enterprise.*

1. Introduction

SOA has been increasingly viewed as an architectural strategy rather than a development approach. The notion of services being a first class objects has sometimes led SOA developers into a trap of realizing web-services without paying a lot of heed to SOA design principles. Although starting from the four tenets of service design by Don Box [9] it has been a much debated issue, the design principles have been at an abstract phase. Aspects such as web-services having autonomous nature and explicit boundaries do not really help a lot in deriving a correct service granularity.

The problem of determining the optimal service granularity has become further important as increasingly applications are being built by assembling internal and external services at a coarser-grain level for both Enterprise Application Integration (EAI) and B2B integration. Since most organizations have overlapping and redundant business functionality and data (e.g., Accounts Management) across different Lines of Business (LOBs), it is essential to have a portfolio of business services which could be reused across multiple places. We take an example of IFBS to show their attempt of such a portfolio design and what typically goes wrong with such an exercise.

1.1. IFBS's business context

IFBS which is one of the world's largest discount brokers serves more than a million clients. Additionally it also offers a range of services likes investment research, mutual funds, annuities, bond trading, and mortgage etc. IFBS has acquired quite a few companies in recent past and their systems have been integrated with its own. IFBS's enterprise architecture (as shown in ***Figure1***) is typical of any large enterprise firm, save the fact that there is a large amount of redundancy in its IT systems, where many systems do the same job with small differences on aspects such as input data etc.

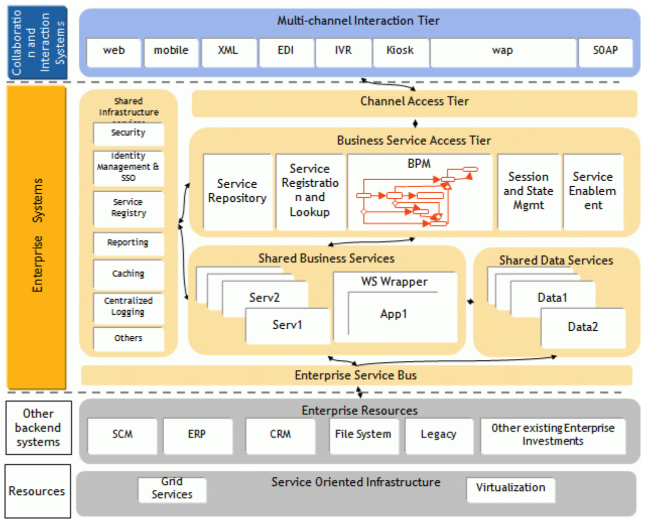
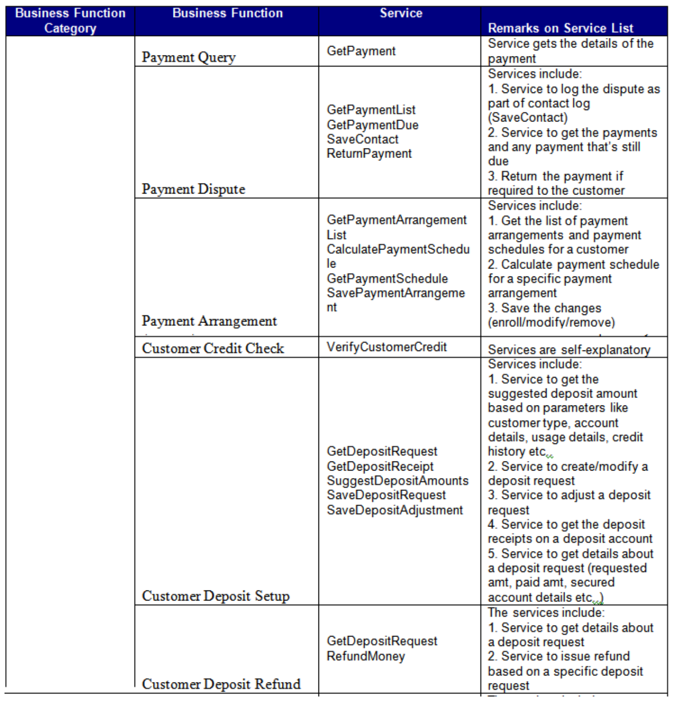


Figure1: IFBS's SOA reference model

Overtime the number of transactions processed by IFBS has increased to hundreds of thousands every day. IFBS proceeds with an exhaustive SOA realization exercise and identifies services for most of its day to day operations. It comes up with a service portfolio (see ***Figure2***) which consists of a large number of services.

[*] *Not the real name (used for reference purpose only)*



Business Function Category	Business Function	Service	Remarks on Service List
	Payment Query	GetPayment	Service gets the details of the payment
	Payment Dispute	GetPaymentList GetPaymentDue SaveContact ReturnPayment	Services include: 1. Service to log the dispute as part of contact log (SaveContact) 2. Service to get the payments and any payment that's still due 3. Return the payment if required to the customer
	Payment Arrangement	GetPaymentArrangementList CalculatePaymentSchedule GetPaymentSchedule SavePaymentArrangement	Services include: 1. Get the list of payment arrangements and payment schedules for a customer 2. Calculate payment schedule for a specific payment arrangement 3. Save the changes (enroll/modify/remove)
	Customer Credit Check	VerifyCustomerCredit	Services are self-explanatory
	Customer Deposit Setup	GetDepositRequest GetDepositReceipt SuggestDepositAmounts SaveDepositRequest SaveDepositAdjustment	Services include: 1. Service to get the suggested deposit amount based on parameters like customer type, account details, usage details, credit history etc. 2. Service to create/modify a deposit request 3. Service to adjust a deposit request 4. Service to get the deposit receipts on a deposit account 5. Service to get details about a deposit request (requested amt, paid amt, secured account details etc.)
	Customer Deposit Refund	GetDepositRequest RefundMoney	The services include: 1. Service to get details about a deposit request 2. Service to issue refund based on a specific deposit request

Figure2: A snapshot showing representative services in the IFBS's large service portfolio

Although on paper IFBS's SOA adoption picture looked rosy, but this whole process made things messy for IFBS. Management of such a huge number of services turned out to be a nightmare, and in-spite of putting a governance framework it was difficult to integrate its services with its business processes. IFBS soon realized that instead of proceeding with a big bang approach of legacy migration to web-services by putting up rappers and partial re-implementation, it required an architectural approach to address the whole issue.

In this work we use the business case of IFBS to drive home the point that mere web-service implementation is not SOA realization. We support our arguments by presenting InSOAP [1] (Infosys Service Oriented Analysis /Adoption Process) and explain how it could be utilized to address the pain points of SOA realization.

The key contribution of this paper is a set of engineering processes and guidelines to address the following key challenges:

Identify candidate services and their relationships
Decide the optimal granularity for services
Decide proper layering and services location within the enterprise

3. Background work

There has been quite a lot of work recently in academia and industry for SOA migration. At Infosys we have been using InSOAP [1] (Infosys Service Oriented Analysis /Adoption Process), an architecture-centric framework, to ease the definition, the design and the realization of SOA. One of the closest works to our approach is IBM's Service-Oriented Modeling and Architecture (SOMA) [2]. SOMA is a methodology for the identification, modeling and design of business aligned services at a proper level of granularity while leveraging existing systems. It helps in the determination of services, flows, and components that realize the services.

Business Applications to Legacy Systems (BALES) methodology as proposed in [4, 5] has been another reverse-engineering based approach to support Web-Services development using "objectified" legacy data and functionality to build business applications. Further there have been other works like by Dabous in [3] who in his work has identified a set of patterns for the architectural design of e-business applications while leveraging functionality or business logic that is embedded in legacy systems. He also proposed quantitative models which can aid in the systematic selection/ranking of patterns in accordance with their appropriateness for a given problem context. Apart from this there has been a lot of work on the SOA realization aspect not all of which are referred in this paper. The focus here is to look at service granularity as a issue and how it can be best resolved. Zhuopeng Zhang et al in [7] provide an overview of service granularity optimization particularly in scenarios dealing with legacy migration.

While most academic works on SOA stress on UDDI based registries, in practice the large scale industrial projects hardly use UDDI's; it has been partly due to their bulk and also the cost of their management. It has been an industrial practice to use governance frameworks which at minimum could comprise of portals for service management. Service granularity thus becomes a key to most of these exercises.

4. Preliminaries

4.1. Defining service hierarchy

The different levels of service granularities in realizing an SCSS are typically process service, business service, Composite Service, Informational service, Data Service, Utility Service, Infrastructure Service and Partner Service. We define these services in **Table1**. While a business service realizes the requirements of the business process where it participates, and is hence IT independent, a data service is the finest grained service which provides operation at implementation level. For an effective SOA realization it is important that the service portfolio takes into account the correct service granularity. Not only it helps in easier maintainability, but it also helps in effective governance of the service portfolios. ***Figure3*** presents a service hierarchy meta-model which is used in some of the heuristic based conversions later in Section 6.

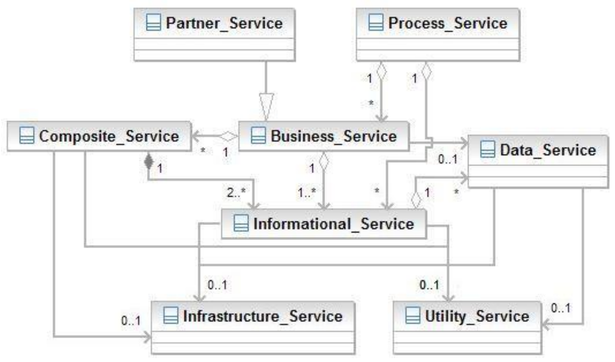


Figure3: Service Hierarchy meta-model

The above service hierarchy normally corresponds to the multi-level process hierarchy as mentioned in [6]. Different levels of services (as in ***Figure4***) can be utilized by different people. For example a business service or an application service can be used effectively by a designer but the corresponding data, utility and infrastructure services have much more importance in application development. An optimal portfolio must hence account for both to promote reusability and ease of maintenance.

Service Type	Details
Process Service	A service whose operations are guided by the process definition. These are reactive services which need business events that would trigger various activities that would be using business and information services.
Business Service	A service encapsulating transactional nature of functionalities that would build business context over other informational service.
Composite Service	A service with either composition or aggregation of multiple other services. The internal invocations are abstracted from the consumer providing a unified view. An orchestration would help composite service to be synchronous in nature and choreography would help composite service to be asynchronous.
Informational service	Services that focusing on providing processed data and whose operations are atomic, executed and realized by one provider on a particular type of runtime environment/platform.
Data Service	Services that provide normalized and aggregated view of critical data entities (or master data) such as Customer, Order, Claim and so on. These services are often realized along with Master Data Management strategies.
Utility Service	A service, whose operations are, shared among various services due to the commonly accepted practices or standardization such as payments, credit card transactions etc. Due to the utility or commodity nature of these services, business might often like to use the best possible provider may be from external sources too.
Infrastructure Service	A specialized technical automation service that provide essential infrastructural capabilities to other services
Partner[†] Service	A manifestation of Business, Informational or Data Service offered to external business partners based on agreed terms.

Table1: Description of service types

[†] Partner service here is treated separately from the other service hierarchy as its granularity can vary from fine grained (as in some external data/function invocation) to course grained (as in external business service usage) depending on the scenario.

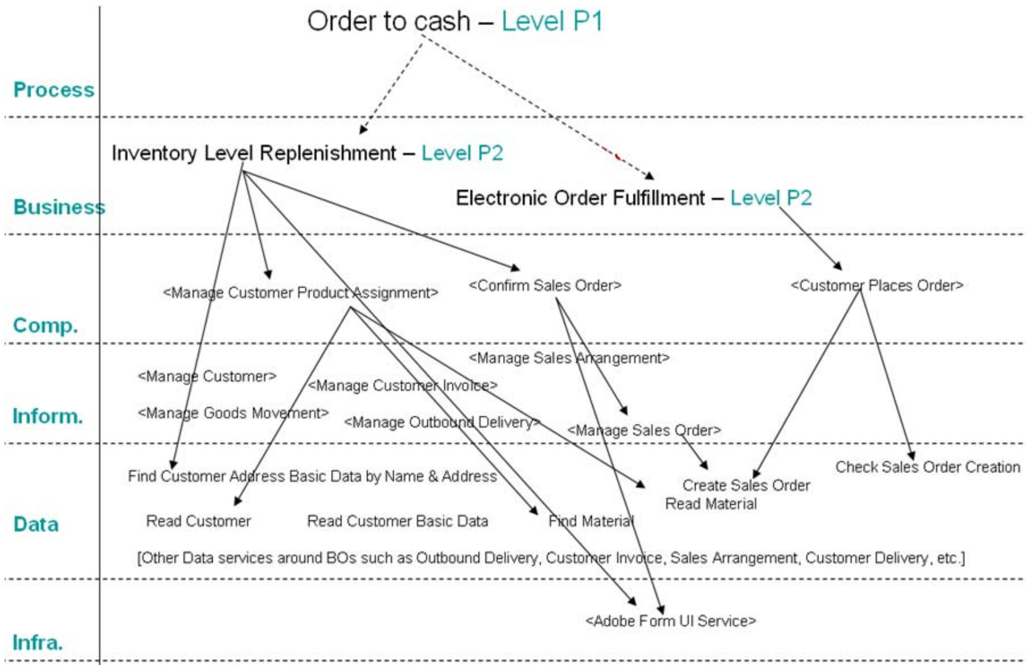

Order to cash – Level P1
Process
Inventory Level Replenishment – Level P2
Electronic Order Fulfillment – Level P2
Business
<Manage Customer Product Assignment>
<Confirm Sales Order>
<Customer Places Order>
Comp.
<Manage Sales Arrangement>
<Manage Customer>
<Manage Customer Invoice>
<Manage Goods Movement>
<Manage Outbound Delivery>
<Manage Sales Order>
Inform.
Find Customer Address Basic Data by Name & Address
Create Sales Order
Read Material
Check Sales Order Creation
Read Customer
Read Customer Basic Data
Find Material
Data
[Other Data services around BOs such as Outbound Delivery, Customer Invoice, Sales Arrangement, Customer Delivery, etc.]
<Adobe Form UI Service>
Infra.

Figure4: Service Types in a multi-tiered process map (Illustrative)

4.2 Service granularity

Services are offered at different layers with a definitive degree of granularity. Service granularity refers to the service size and the scope of functionality a service exposes. The service granularity can be quantified as a combination of the number of components/services composed through a given operation on a service interface as well as the number of resources' state changes. The service should have the right granularity to accomplish a business unit of work in a single interaction.

A service would be regarded too coarse-grain if the size of exchanged messages grows and sometimes might carry more data than needed, or presents a complex interface which is prone for regular changes. On the other hand if the service is too fine grained multiple round trips may be required which would introduce quality concerns and outflow of services. A balance is hence required between level of abstraction, likelihood of change, complexity of the service, and the desired level of cohesion and coupling. A tradeoff needs to be made while taking into account non-functional requirements particularly performance.

4.3 Key issues arising out of improper service granularity

Some of the issues which can arise out of improper service granularity are as follows:

- Service Duplication (Different services for similar tasks)
- Difficulty in maintenance
- Service Governance is extremely difficult when the number of services is huge and the services are merely fine grained interfaces
- Service reuse across applications suffers defeating the basic fundamentals of SOA
- Business and technology alignment gets extremely difficult leading to redundancy within enterprises
- It gets difficult to assign SLAs and KPIs for individual services and thus audit operations become almost impossible

5. Using INSOAP ADM approach

InSOAP provides a systematic approach and a well-defined process to guide the design, evaluation and development of a Service Oriented Enterprise Architecture. InSOAP comprises of four phases as depicted in ***Figure 5***. The goal is to define and realize an enterprise-wide SOA. We advocate this approach as it yields more cohesive and uniform enterprise architecture, and reduces redundancy, thereby lowering development and maintenance costs. It also allows the business to identify more reusable shared services and address issues such as shared access to information such as a single view of the customer.

IFBS choose to relook at their service portfolio as well take up a strategic view by adopting InSOAP. The systematic approach of InSOAP provided an opportunity for clearly defining the goals for service enablement and brought out a discipline within the organization on how services are created or consumed.

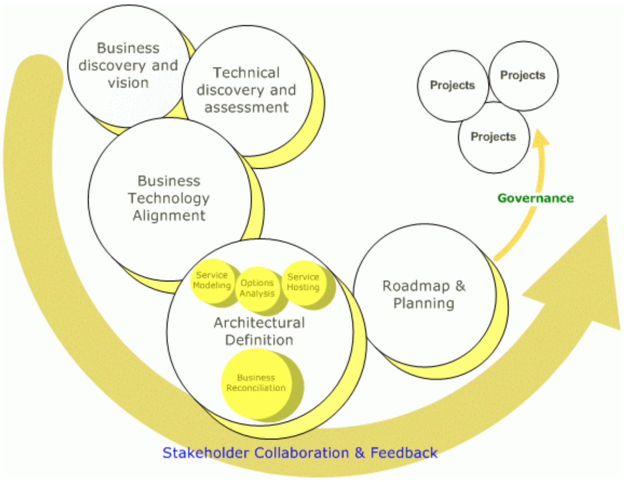


Figure5: InSOAP phases mapped to key TOGAF deliverables

5.1 Reevaluation of IFBS's service portfolio

The present architecture which was cluttered with many rouge services was relooked during the 'Technology discovery and assessment' phase. The primary objective in this phase was discovering the usage of the current services and mapping them to business processes as well as existing IT systems. This process is referred to as "Process-To-Application" (P2A) and "Application-To-Services" (A2S) mapping. This mapping provided an insight into the services which were catering to business activities and operational applications. The result of such mapping which was captured in a simple spread sheet highlighted redundancies and overlaps in the current service portfolio.

An analysis was conducted over the current service portfolio to identify their suitability for layering and categorization at a later stage. The analysis included the following criteria:

- **Business value** a criterion which is associated with the benefit that an organization as a whole realizes in economic terms. This essential would involve cost benefit analysis or in other words return on investment (ROI). IFBS usually opted towards guess work for realizing the value of building a service. Hence while calculating the business value for services an ROI analysis was resorted to that was based on simple approach. ROI of services were calculated by identifying various costs that were incurred in building a services and the benefits that IFBS has seen since the services were made available. The cost parameters included resources allocated for services (inclusive of new infrastructure and software/products) and effort (inclusive of building, maintaining and governing). On the other hand benefit parameters considered were percentage reduction in - time taken to build new applications, cost incurred on building applications, time taken for integration, cost of integration effort, cost of maintenance and percentage increase in – number of new solutions/applications built, number of services reused across multiple applications, operational efficiency achieved and application of compliance requirements. Based on these parameters the ROI for each services were calculated. A fairly simple summation of the costs and benefits was used for calculating business value was used. It was also considered that a higher value indicated the higher risks involved with any realignment or transformation attempt.

- **Functional reusability** was used to measure the ability of the services to provide a generalized set of services, compared to the development of a specific service for a specific consumer application. As shown in **Figure 2** that portfolio consisted of many granular services which were result of immediate need. Such services were inadvertently added into portfolio resulting in duplication of functionalities amongst many services. Increased reusability stems mainly from accurate, complete and generalized service contract design capturing all possible message variants. This allows covering a larger number of usage scenarios through altering the service behavior simply by supplying varying message instances. A message instance will confirm to a subset of a super-schema defined by the service contract. So number of message instances that a service can support was considered as a measure for functional reusability.

Another measure that was considered during evaluation of reusability was the ability to be able to compose or in other words design of service for assembly. It is important that a service interface is defined in a way that its encapsulated functionality can be used and composed in different contexts with minimal effort so as to increase the service reuse potential. It was seen that IFBS services were directly exposed out of existing systems. Hence most of the services in the portfolio were non optimal and needed substantial effort to aggregate. Direct exposing of underlying components as services resulted in complicated interface which were hard to comprehend. So number of interfaces that a service supported and the ability for users to understand the interface was a measure which was considered.

It was found from IFBS case that since the use was unplanned there was more dependency rather than reuse. This introduced risk of breaking a service or application during realignment.

- **Technical health** of services included evaluating soundness of the approaches taken to realize service. Various approaches can be taken to realize services. This can be broken in two categories – strategic and tactical. Strategic approaches include transformation while enabling services. It involves a deep and detailed analysis of the existing code base, understanding the system functionality and data architecture. Subsequently, it involves the extraction and rationalization of data definitions, data and business rules. This is followed by an iterative process that involves refactoring, consolidation, componentization and redesigning activities to make the code more modular and ease the incremental migration to a flexible architecture. Where as a tactical approach will be wrapping of existing assets in a standard format. This requires less up-front architecture and design investigation. This can only provide a tactical short term solution as it addresses the integration and flexibility pain points without impacting the source code significantly. Hence the measure chosen to evaluate technical health was approach (method and technologies) taken to enable service from the existing assets.

- **Technical flexibility** measured the level of service complexity and extensibility in terms of technologies used and product dependencies. This essentially indicates how easy it is to modify or extend the service in the case of service reengineering. As the demand for service enabling organizations are increasing so is the infrastructure needed to support the services. Variety of new technologies have emerged to support services such as Enterprise Service Bus (ESB), Web Service stacks, products that enable services from components and legacy assets, Service Data Objects (SDO), registries, etc. Amongst these new developments, primary importance was given to the adoption of web services standards both core and WS-*. Even the various products used were evaluated against the web services standards.

  Hence the measure for evaluating technical flexibility included the availability of human resources to manage services, technology and standard adoption for services, popularity of the products chosen for services.

5.2 Service architecture strategy for managing services

A detailed architectural blueprint was defined for managing service based on their SOA model (refer ***Figure1***). The previous phases provided the detailed input for the definition of the architectural plan. The inputs include pain points related to creating, developing and consuming services, analysis of the current service portfolio, and mapping from P2A and S2A. This blueprint provided IFBS a well defined approach and guide to the evaluating the need, designing, developing and consuming of services in future. The architecture was defined three levels of abstraction in conceptualizing services – enterprise level, line of business level and project level.
At enterprise level, some strategies included a light weight governance model and service portfolio management. Line of business level architectural strategies included service value mapping, reuse guidance, and defining the initiatives in accordance to technology roadmap. The project level strategies included usage guidance for technologies, products and standards.

5.3 Consolidating services through classification and layering

Consolidation of services is an iterative process for arriving at an optimal services composition. The aim is to first establish clear and well-defined boundaries between collaborating systems, followed by reduction of interdependencies and limiting of interactions to well-defined points. The key tasks in the process include identification of services along with deciding service granularity and appropriate layering of services.

The service identification process required domain analysis and decomposition to identify valuable and reusable business functions to be provided as services. The identified services needed their relationships to be rationalized and consolidated with existing set of services. In order to do this the stakeholders were involved to agree on the high level service definitions that addressed business technology goals. It has been our experience through the development of the case study that the service as an architectural entity is subjected to vary due to the various frames of thought. It was also found that there could be multiple objectives for having to realize the services such as making certain business functions as commodities, integrating partners, bring agility to embrace dynamic market conditions etc.

Following identification, classification of services is an important. Classification of services is essential in understanding of granularity of services and hence their compose-ability. New portfolio of services for IFBS was classified both horizontally and vertically. Horizontally services were classified based on the hierarchy as described in Section 4.1. By classifying services horizontally the principles of services interactions can be enforced. Further, this type of classification helped in arriving at the requirements for non-functional aspects of services design, for example core and common services need to be designed and deployed with more emphasis on scalability and high availability. Vertically services were classified following a taxonomy based on the business domain such as investment management, brokerage, retirement planning, taxation, annuities etc. This classification assisted business to understand the reusability while they explore the possibility of new offerings.

As described in Section 4.2 granularity of services was defined based on the assumption that different layers would have different granularity. We employed model based approach to solve the problem of achieving the proper level of granularity of services. As throughout the service identification the services identified were in conformance with the model and inherently enforced the relationships among services as defined in the model. ***Figure6*** shows a model which is derived out of the service hierarchy meta-model as described in Section 4.1.

Using the model based approach for arriving at granularity we identified preconditions for each relation within the model. Some of them are as follows:

- The aggregations defined a strict boundary between the services.
- A composition indicated that services can be coarser grained either through orchestration or choreography.
- Using Generalization tries tighter boundaries can be defined
- A Usage relationship indicates that services are loosely coupled and have least dependency.
- A Dependency relationship insists that there would be a need to have a strict guidance on service usage.

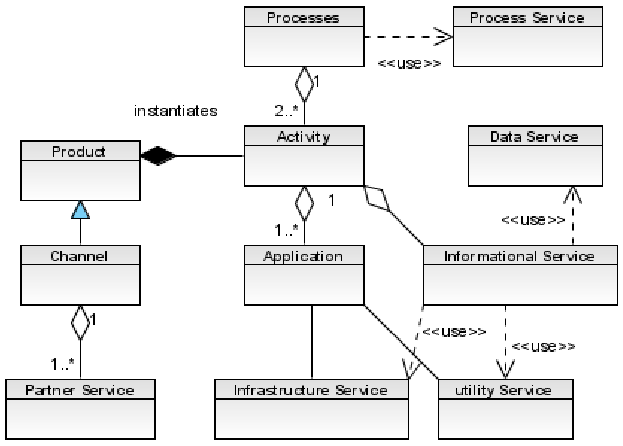


Figure 6: Service model to derive granularity

Some of the results from the above exercise include deriving hierarchical services from the existing service portfolio. Using aggregation relationships we were able to define process and composite services along with the fine grained services.

6. Conclusions

It has been our experience that creation of web services in itself doesn't really help to deliver increased reusability, flexibility and responsiveness to change. This requires a strategic approach towards building a strong service foundation as well as sound engineering principles to realize it. An important advantage we derived out of this approach was that the relationship based granularity clearly endorsed the necessity of coarser grained services to adhere to the single service view point and fine grained services was optional.

8. References

[1] Abdelkarim Erradi, Sriram Anand, and Naveen N. Kulkarni: SOAF: An Architectural Framework for Service Definition and Realization. IEEE SCC 2006: 151-158

[2] Ali Arsanjani, Abdul Allam: "Service-Oriented Modeling and Architecture for Realization of an SOA". IEEE SCC 2006: 521

[3] Dabous, F. T. 2005, *A Pattern Based Approach for the Architectural Design of e-Business Applications*. PhD thesis, School of IS, UNSW, Sydney, Australia.

[4] Heuvel, W.-J. v. d. 2004, 'Matching and Adaptation: Core Techniques for MDA-(ADM)-driven Integration of new Business Applications with Wrapped Legacy Systems', in

Model-Driven Evolution of Legacy Systems (MELS'04), Monterey, Canada

[5] Heuvel, W.-J. v. d., Hillegersberg, J. v. and Papazoglou, M. P. 2002, 'A methodology to support web-services development using legacy systems', in *IFIP Working Conference on Engineering Information Systems in the Internet Context*, Kanazawa, Japan, pp. 81-103.

[6] J. M. Bhat and N Deshmukh, "Methods for Modeling Flexibility in Business Process", BPMDS 05 Proceedings, June 2005

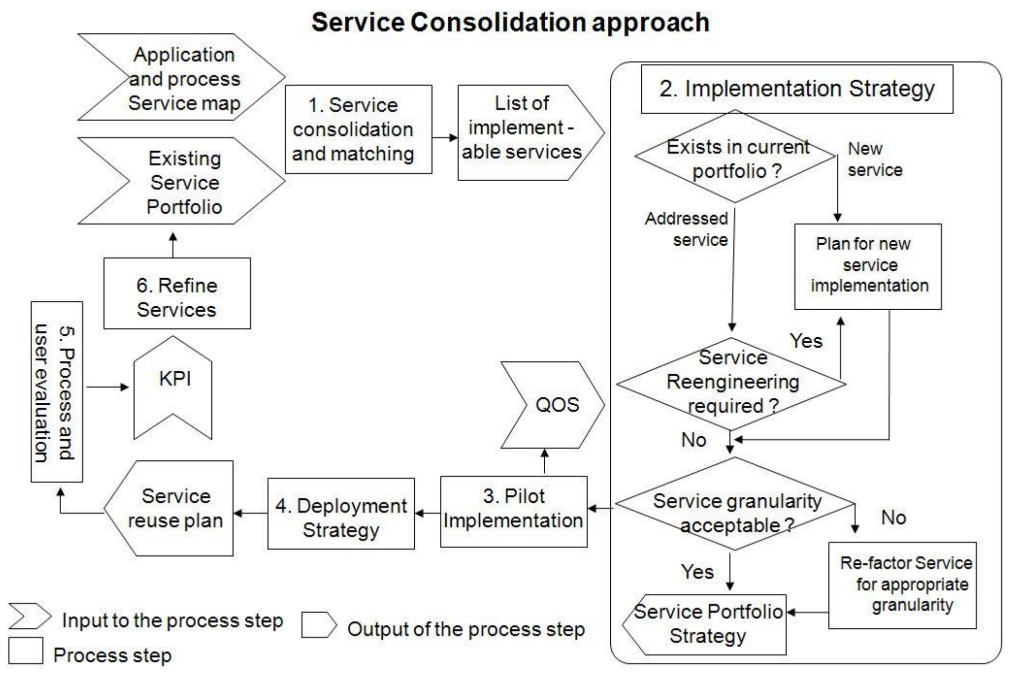
[7] Zhuopeng Zhang, Ruimin Liu, Hongji Yang:, "Service Identification and Packaging in Service Oriented Reengineering" SEKE 2005: 620-625

[8] Abdelkarim Erradi, Naveen N. Kulkarni, Piyush Maheshwari: Service Design Process for Reusable Services: Financial Services Case Study. ICSOC 2007: 606-617

[9] Don Box, "Service Orientation and Its Role in Your Connected Systems Strategy"

http://msdn2.microsoft.com/en-us/library/ms954826.aspx

Appendix


Service Consolidation approach
Application and process Service map
Existing Service Portfolio
1. Service consolidation and matching
List of implement-able services
2. Implementation Strategy
Exists in current portfolio ?
New service
Addressed service
Plan for new service implementation
Yes
Service Reengineering required ?
No
Service granularity acceptable ?
No
Yes
Re-factor Service for appropriate granularity
Service Portfolio Strategy
QOS
3. Pilot Implementation
4. Deployment Strategy
Service reuse plan
5. Process and user evaluation
KPI
6. Refine Services
Input to the process step
Output of the process step
Process step