Table of Contents

1 THE BUSINESS CASE ................................................................................................................. 3
  1.1 OVERVIEW .......................................................................................................................... 3
  1.2 BUSINESS AND IT OBJECTIVES OF THE MIGRATION ....................................................... 4
  1.3 PROJECT SCOPE ................................................................................................................. 5

2 THE SERVICE ORIENTED ARCHITECTURE ....................................................................... 8
  2.1 SERVICE LAYERING .......................................................................................................... 8
  2.2 CAREY’S SERVICE ORIENTED ARCHITECTURE ............................................................... 10
  2.3 SAMPLE SERVICES ............................................................................................................ 11
  2.4 LEGACY SYSTEM WRAPPER ........................................................................................... 13

3 CONCLUSIONS .......................................................................................................................... 14

List of Figures

FIGURE 1 THE "ACCIDENTAL ARCHITECTURE" ................................................................... 4
FIGURE 2 TARGET SYSTEM OVERVIEW ............................................................................... 7
FIGURE 3 SERVICE LAYERING .............................................................................................. 8
FIGURE 4 CAREY’S SERVICE ORIENTED ARCHITECTURE ................................................... 10
FIGURE 5 LEGACY SYSTEM WRAPPER ................................................................................. 13

List of Tables

TABLE 1 BUSINESS OBJECTIVES ADDRESSED BY IT ......................................................... 5
1 The Business Case

1.1 Overview

Carey is one of the world's leading chauffeured services companies. Their clientele comprises individuals (including many celebrities), corporate clients, as well as travel agencies. Carey’s value proposition is based on delivering a highly personalized level of service and mission-critical reliability, with a high level of consistency and safety.

Carey provides services to its customers via Carey owned subsidiaries, affiliates, licensees and farm outs (i.e. agreements with third party providers). The IT environment at Carey subsidiaries consists of locally installed and managed applications. Those include legacy software (“CSW”), which manages reservations, dispatches jobs to chauffeurs, and handles end-of-job functions. Some of the subsidiaries and franchise operations use the “Hudson” software, which is a third party system that provides basically the same functionality as CSW, but in addition it also includes billing functions. Besides these core systems there are a number of other applications that are mostly provided by third party vendors.

The IT environment at Carey corporate headquarters consist of CESRes, a homegrown application that manages reservations, profiles and customer service functions. It was originally developed as a Java/CORBA application and was later migrated to a J2EE/WebSphere platform. It supports multiple inbound channels, i.e. the Carey Web site and call centers that utilize rich clients (Swing based). The other core systems at the corporate site include a legacy, 4GL based application called “Friendly Billing”, which handles generation of invoices, and a Oracle/PeopleSoft package that performs all the AR/AP functions.

In addition to the IT systems that run at the subsidiaries and at Carey headquarters, there are external sources of business information, e.g. Global Distribution Systems (GDSs) that feed reservation requests from travel agencies and other partners to Carey.
1.2 Business and IT Objectives of the Migration

The multitude of applications that are deployed at the subsidiaries as well as the inflexible legacy systems that run in the Carey headquarters have grown into a “accidental architecture” that has hampered the business. Specifically, about 80% of reservations are made at one of the local offices and only about 20% reservations are made at the call center or via the company web site. This creates a huge integration problem and a lot of manual labor especially when it comes to handling exceptions. Figure 1 gives a schematic overview of this architecture.

![Figure 1 The "Accidental Architecture"](image)

The broad goals of the migration to a Service Oriented Architecture (SOA) are:

- Support all of the core business functionality from one (logically) central location
- Support the expansion of service offerings
- Support the acquisition of other companies
Table 1 provides a detailed list of Carey’s business objectives and maps them to the specific IT approaches that have been employed in order to address them.

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Addressed by IT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue Enhancement</td>
<td>Better leverage of the IT infrastructure and processes</td>
</tr>
<tr>
<td>Cross-selling</td>
<td>Increased extensibility and adaptability of the core applications</td>
</tr>
<tr>
<td>Acquisitions</td>
<td>Avoid one-off integration solutions</td>
</tr>
<tr>
<td>Customer Service Enrichment</td>
<td>Creation of a centralized reservation system with customer profiles</td>
</tr>
<tr>
<td>Single view if the customer</td>
<td>SOA enables efficient integration of a portal with back office systems</td>
</tr>
<tr>
<td>Better customer self-service</td>
<td>Centralized SOA leads to improved business process consistency</td>
</tr>
<tr>
<td>Improved business process consistency</td>
<td></td>
</tr>
<tr>
<td>Total cost of ownership (TCO)</td>
<td>Service reuse, decreased complexity</td>
</tr>
<tr>
<td>Risk management</td>
<td>Core services (foundation and some application) are implemented by highly skilled designers/developers</td>
</tr>
</tbody>
</table>

Table 1 Business Objectives Addressed by IT

1.3 Project Scope

The overall goal has been to implement a central system that delivers full business functionality to all Carey sites, while being maintained in a single location. The scope of the project includes in particular:

1. Consolidation of the reservation system

   - A wholesale re-implementation of the legacy reservation system was considered to be too costly and introduce too much risk in the first phase of the project. The company decided for a strategy that consisted of migrating the legacy Java/CORBA code to the new J2EE environment, encapsulation of existing functionality as services where possible, and surrounding the legacy application by new services, such that the functionality of the legacy system could gradually be decommissioned.
2. Replacement of the billing system
   - The new billing system should replace the central legacy 4GL system (i.e. “Friendly Billing”), local systems installed in some subsidiaries, and streamline the billing process for a number of subsidiaries that used a manual process to transfer billing to the central system. In addition, it would allow chauffeurs to send adjustments of the originally quoted price directly from their handheld devices to the central system.

3. Replacement of the local dispatching applications
   - All dispatching application that had been deployed locally in various subsidiaries should be replaced by a new central dispatching system.

4. Implementation of a SOA foundation
   - The migration of the reservation system and the re-implementation of the billing system and the dispatching applications was the primary goal of the project. However, the company wanted to capitalize on the opportunity and start building a SOA foundation that consists of common infrastructure services and common application services, i.e. services that can be reused across several business processes.

Figure 2 shows an overview of the new applications (depicted to the right) and the different user constituencies (to the left) that can access the new SOA-based business processes through a portal or through a B2B Gateway.
The new applications implement a variety of business processes, whereby each business process is composed of a number or reusable services. The services are organized in three categories (orchestration, application, and infrastructure) and one could envision a matrix where the applications are the columns, and the services are the rows of the matrix. This would visualize how the services are utilized across the applications (i.e. business processes).

The business processes are made available through a portal to various user constituencies. These including booking agents in calls centers who are responsible for dispatching the cars, the chauffeurs who need to receive jobs and provide status information back to the central site, some large customers who can access select business processes, etc.

Some business processes are also exposed through a B2B Gateway to business partners, including electronic reservation systems and partners (i.e. sub contractors or affiliates) who provide chauffeured car services on behalf of Carey.
2 The Service Oriented Architecture

The SOA has been developed following best practice guidelines for service analysis, modeling, and design. Key architectural principles that were applied are the separation of concerns and design for reusability. This is achieved by a separation of functionality into layers.

2.1 Service Layering

The breakdown into distinct layers facilitates decoupling of the services. A typical service layer model for SOA is comprised of an orchestration services layer, an application services layer, and an infrastructure services layer. This is illustrated in Figure 3.

The top layer of the SOA is comprised of orchestration services. An orchestration service acts as a controller, composing application services and infrastructure services to implement a business process. For the most part, an orchestration service is made up of workflow logic and calls to lower level services. This allows orchestration services to be changed and adapted to new business process requirements without affecting the underlying application and infrastructure services.
The middle layer is comprised of application services that implement business logic. One of the most important considerations when designing application services is to make sure they are reusable across different business processes.

There are typically two implementation approaches: one can develop an application services from scratch, or exploit the functionality of a legacy system or packaged application by wrapping. The latter requires building a “wrapper” that encapsulates the existing system such that it becomes compatible with the new service oriented architecture. This is often not as simple as it may sound, which will be illustrated later in this case study.

The bottom layer of the SOA is comprised of infrastructure services. They contain technology specific functionality (e.g. sockets communication, persistence service) and do not include business logic.

Below the three layers of the SOA we depict components and legacy systems. The red arrows indicate that application services and infrastructure services either are implemented through components to perform some business or system logic, or they invoke functionality that resides in legacy systems.
2.2 Carey’s Service Oriented Architecture

These architectural guidelines were applied to the implementation of Carey’s Service Oriented Architecture, which is illustrated in Figure 4.

The Carey SOA adheres to the separation into the three layers of services. The application services layer also contains wrapper services that encapsulate legacy systems (e.g. the old reservation application, “CESRes”) and packaged applications (the Oracle/PeopleSoft package). Figure 4 only shows a few of the major business processes in the orchestration services layer.

The three core applications (Dispatch, Billing, and Reservations) are depicted as business processes. This should emphasize the changing notion of what constitutes an application. Applications used to have well-defined boundaries of code, i.e. the application “owned” the code. This had an important implication on how software reuse could be achieved and managed: when objects were reused: it basically meant that the object code was copied into a new application.
In case of a SOA the notion of an application and its boundaries changes drastically. Service oriented applications are really a composition of services into business processes. The code of the services is not copied into a business process – the process rather uses a run-time instance of a service. All three business processes that have been implemented, i.e. Dispatch, Billing, and Reservations, use several of the application and infrastructure services in different combinations.

It goes beyond the scope of this case study, but it is generally helpful to organize all services into a service taxonomy. The separation into the three layers is part of this taxonomy. In addition, services could be classified by the business domain they belong to, for example billing or reservations (in large enterprises the Line of Business (LOB) boundaries would provide another classification). Services that can be used across business domains or LOBs should be classified as enterprise services.

If applied properly, such a service taxonomy can be an essential tool for governance of the service development lifecycle. When a developer requires a service for a new business process, the taxonomy would guide him in looking for existing reusable services. If the service does not already exist, the developer would have to consider for what category the service should be developed, which would clarify the participation of different groups in the requirements analysis and design process. It can also help to clarify the functional boundaries of the service such that it does not include capabilities that extend outside of its category. Finally, the service categorization would impact the maintenance responsibilities for the service.

2.3 Sample Services

Numerous services have been implemented to fulfill the requirements of the three core business processes and to build a foundation of reusable enterprise services. Examples of infrastructure services include:

- **Persistence Service** – this is a data access service that insulates the database from the various applications or services that need to persist state; this approach avoids the problems that occur when each application group writes their own data access code: some may use SQL, others stored procedures, Hibernate, TopLink, JDBC, etc. In addition, unmanaged data access often leads to freezing the database schema due to the dependencies introduced by multiple data access methods. The Persistence Service works with Hibernate to map between the application object model and the relational structure of the database.
• **Security service** – any application that is made available across a wide area network should have a security model in place to keep out unwanted users and to limit users to the views and operations that are specific to their roles. The rules that govern security are general enough that they will apply across all application domains within Carey and therefore are realized in a single security service. The security service provides two main functions, authentication and authorization. Authentication determines whether the user has access to the application via ID/Password, and in case of external partners also through certificates. Once authenticated, the user needs to be authorized to execute a predefined function.

• **Transformation Service** – The SOA introduces the concept of a common, XML based format for all data that flows through the entire system. The common format reduces the number of point to point transformations that need to be built and maintained. Each application only needs to understand how to map from its own format to the common format; it is not concerned with the proprietary formats that other services employ. Another advantage is that the application development teams can implement solutions to a certain degree independent of each other. The Transformation Service converts data between the common XML representation and legacy or proprietary formats.

There are a large number of domain specific application services. We list some of them here briefly:

• **Account Service** – allows to manage customer accounts;

• **Passenger Service** – allows to manage customer profiles, i.e. their contact information, travel preferences, etc;

• **Provider Service** – some jobs are farmed out to sub contractors. The Provider Service allows to manage their chauffeurs, vehicles, vehicle types, costing tariffs, etc;

• **Rate Service** – allows to manage rate schedules, including both pricing rates and costing rates;

• **Job Service** – allows to manage jobs and all business behaviors associated with a job, e.g. job status changes, collecting end of job information, adjusting pricing and costing, etc;

• **Staging Service** – bulk data arrives from a number of different sources. These sources include “end of job” files from affiliates, travel agent information, and currency exchange rates. This data must be captured, stored, and made available within the domain object model;
2.4 Legacy System Wrapper

As mentioned above, a wholesale re-implementation of the legacy reservation system was considered to be too costly and introduce too much risk in the first phase of the project. The company decided for a strategy that consisted of migrating the legacy Java/CORBA code to the new J2EE environment, encapsulation of existing functionality as services where possible, and surrounding the legacy application by new services, such that the functionality of the legacy system could gradually be decommissioned.

It is often the case that middleware vendors (e.g. Enterprise Service Bus) gloss over the legacy integration issues, stating that it is simply a matter of building a wrapper around the legacy system such that some of its functionality is exposed as services. However, many legacy systems do not have well defined APIs that can be invoked through a function call, or accept messages based on message oriented middleware.

The legacy reservation system in this case study represents an example of such a situation. The presentation logic that supported the Web front end was not cleanly separated from the business logic. In order to create a new service oriented interface it was therefore necessary to use an invasive approach, in other words the code of the legacy system had to be modified.

The presentation logic had to be removed and the business logic isolated such that the functionality of the reservation system could be exposed in service oriented APIs. This is illustrated in Figure 5:

![Figure 5 Legacy System Wrapper](image-url)
As a result, there was only one API that all types of clients could utilize to interact with the reservation system. The different types of clients include a Web client, the Swing client of a desktop application, any of the B2B channels that are part of the B2B Gateway (e.g. the GT3 channel and the CSI channel), as well as affiliates that access the reservation system over the company’s wide area network (e.g. the Hudson system).

3 Conclusions

The success of a SOA project depends to a large degree on service reusability. The potential for reusing a service is significantly improved when the SOA follows a proper layering of the services, i.e. orchestration, application, and infrastructure services. The application and infrastructure services in this case study provided business process agnostic functionality that could be reused across three different business processes (dispatch, billing, and reservation).

Reusability doesn’t just happen; it necessitates a development process that is geared towards a service oriented approach. This can be achieved by adapting a traditional development methodology (e.g. RUP) to SOA. In addition, an initial investment into the architecture is required, in other words, the overall SOA needs to be defined first before implementing a particular project.

One has to take into account that the first architected sub-project does not benefit from reuse, since it contributes services for later reuse. Although the cost savings of service reuse can easily be demonstrated, developing for reuse incurs additional upfront cost. A successful SOA requires a long term plan that includes budgeting for reuse. However, business pressures have sometimes forced a shorter term view on the project, which was less effective in the long run.

It is also important to be cognizant of the culture of the IT organization. The client’s development organization had been specializing in J2EE and Spring, which meant that they were very focused on a component based approach to application development. Therefore, moving to SOA required a paradigm shift and was sometimes met with resistance. This was eventually resolved and a by-product of the SOA projects is a development organization that has assimilated new skills and expanded their tool chest.