Middleware introduction pitfall

J. de Baat
30-07-2007

Table of contents

1 Introduction 3

2 Four Layer Model 4
2.1 Environment layer 4
2.2 Business Architecture layer 5
2.3 Application Architecture layer 5
2.4 Technical Architecture layer 6

3 Mobile operator example 7
3.1 Introduction 7
3.2 Infrastructure architecture 7
3.3 Application architecture 8

4 Middleware introduction 9
4.1 Eliminating point-to-point interfaces 9
4.2 Infrastructure architecture 9
4.3 Application architecture 10

5 Functional pitfalls 11
5.1 Different methods 11
5.2 Different functional levels 12

6 Conclusion 14

Table of figures

Figure 1 The Four Layer Model 4
Figure 2 A mobile operator example 7
Figure 3 Infrastructure architecture 8
Figure 4 Application architecture 8
Figure 5 Introduction of middleware 9
Figure 6 Middleware infrastructure architecture 10
Figure 7 Middleware application architecture 10
Figure 8 Functional pitfall regarding different methods 11
Figure 9 Functional solution regarding different methods 12
Figure 10 Functional pitfall regarding different functional levels 12
Figure 11 Functional solution regarding different functional levels 13

1 Introduction

The main benefit of the introduction of middleware is to reduce the interface complexity of the existing point-to-point connections between the different applications. However, the return on investment is often less than expected. This article presents a pitfall which seems obvious but is seldom avoided.

Chapter 2 introduces the Four Layer Model which is used as theoretical background. As an example often works best to explain the situation, chapter 3 introduces a simple environment based on a telecom operator business with only 4 systems. The typical introduction of middleware for this example is explained in chapter 4. As the previous chapters introduce the situation, chapter 5 handles the pitfalls this article is about. Chapter 6 finally presents the conclusion that can be drawn.

2 Four Layer Model

In our view, the context of architecture can be viewed as a set of four layers as shown in Figure 1. This figure shows a model of the organisation and its relation to its environment. The layers which make up the architectural context, have a direct relationship with the layers directly above and below, as indicated by the arrows. Each layer should provide a predefined set of services to the layer directly above so that this higher layer can use these services in its own functionality. The exact services requested are to be described in a set of requirements which the higher layer poses onto the lower layer.

An often occurring fault is that the layers Environment and Business are still ill-defined when far reaching decisions in the Infrastructure layer are being considered. The model shows that requirements and expectations of the Application layer must be based on the Business layer. IT should follow business requirements. Architecture and the architecture process must ensure that this relationship remains intact.

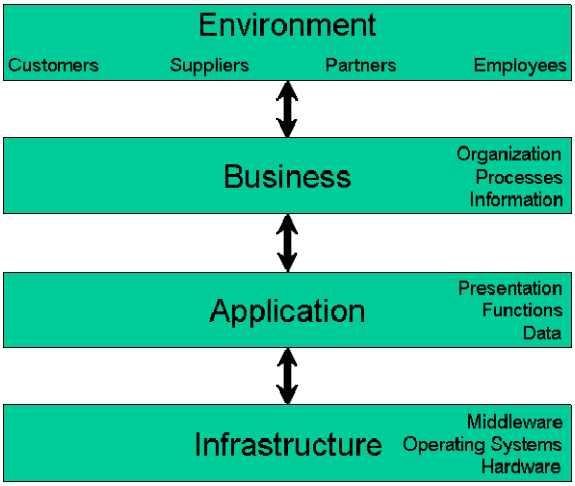

Figure 1 The Four Layer Model

2.1 Environment layer

The Environment layer contains the complex of all external entities to which the internal business processes are related. It is also sometimes referenced by "the market" or market sector. In this figure the environment consists of Customers, Suppliers, Partners and Employees.

2.2 Business Architecture layer

The business architecture layer describes the structure and co-operation of the business processes and the necessary information as supported by the application architecture. It also gives structure to the products and services in a way the business could understand. The business architecture layer itself can be divided into three sub-layers as shown in Figure 1:

- **Business Organization**
  Business Organization describes the structure and contents of the products and services of the business for its environment, the market. This determines the functionality and the requirements which are demanded by the environment, e.g. the structure of insurance products and other financial services.
- **Business Processes**
  The Business Processes layer describes how the business functionality is specified and deployed to provide the desired services to the layer above. To this purpose, the processes pose some requirements to the necessary information.
- **Business Information**
  The Business Information layer describes the structure and definition of the information (data) and their form which is necessary to support the Processes.

2.3 Application Architecture layer

The application architecture layer is a translation of the business architecture into a set of co-operating applications, mostly based on components, running on the implementation of the technical architecture. The application architecture contains the set of models which describe the different applications necessary to provide the required functionality to the business. Again we distinguish three more layers on this level:

- **Presentation**
  The Presentation layer contains the functionality to communicate with the end-user of the application. Mostly, this will be a Graphical User Interface (GUI), in older systems, a Character-based User Interface (CUI) is used.
- **Business Logic**
  The Business Logic layer contains the actual business rules and other business processing functionality to provide the services required by the business architecture.
- **Data Access**
  The Data Access layer describes the systems which are used to get access to the actual business data. This layer deals with the functionality related to database access and data persistency.

2.4 Technical Architecture layer

The technical architecture layer is a description of the actual systems, in terms of hardware, operating systems and middleware, the application components should be deployed on. Note that this includes all systems on all locations. The Technical Architecture describes the actual infrastructure which is used to deploy the components presented in the application architecture. As most good things, this architecture also contains three more sub-layers:

- **Middleware**
  The Middleware layer describes how the different systems are connected to provide a single (virtual) system to deploy all applicational components on. In modern architectures, the term middleware is also used to indicate the implementation of services which are capable of hiding the specific features of the different systems comprising the total technical infrastructure.
- **Operating System**
  The Operating System layer determines for each different piece of hardware which OS it runs and which functionality it can provide.
- **Hardware**
  The Hardware layer explicitly states which iron is available to be used by the applications. This comprises both the physical computers as well as the network components like cables, routers, hubs and the like.

3 Mobile operator example

The Four Layer Model presented in the previous chapter is very useful as a reference for developing systems in a consistent and incremental way. In this article, we concentrate on the two lower layers, Application and Infrastructure. This chapter describes the example of a simple environment typical for a mobile operator.

3.1 Introduction

Figure 2 shows the example which consists of two back-end and two front-end systems. The first back-end system provides the Billing functionality and the second back-end system is used as a CRM system to manage the customer relations. Both back-end systems can be reached by customers via either the internet (Web) or phone (IVR).

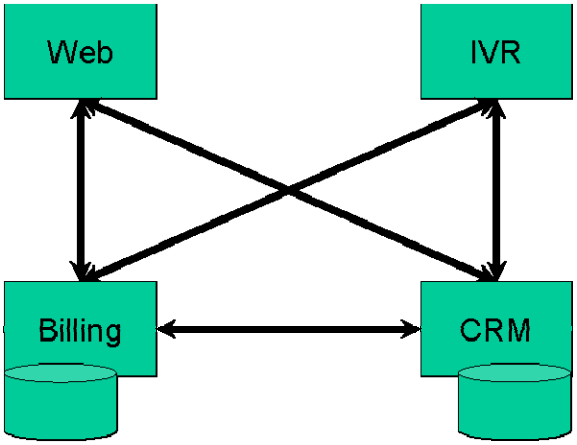

Figure 2 A mobile operator example

3.2 Infrastructure architecture

Figure 3 shows the infrastructural aspects of the example. For the purpose of the example, all 4 systems run on different platforms and different environments. Both front-end systems run on light weight platforms like Windows 2003 and Linux. The communication with the back-end systems runs over TCP/IP. The protocols used on higher levels however are also different, e.g. SOAP, RPC. The back-end systems run on mid range platforms like HP-UX or IBM OS/390. This diversity of platforms is often the first reason to implement middleware.

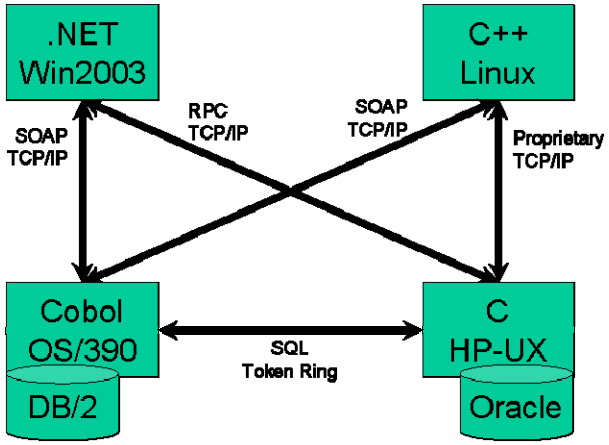

Figure 3 Infrastructure architecture

3.3 Application architecture

The application architecture for this example is shown in Figure 4. In its simplicity, the Billing system provides information on invoices, based on the CDRs stored in its database. The CRM system stores the service settings for customers. Both the Web and IVR front-ends enable the customer to update the service settings and get information on the invoices of the customer. In addition, the Web also provides a simple e-shop for new customers; the resulting orders are processed by the Billing system, which in turn updates the CRM system with the new customer data.

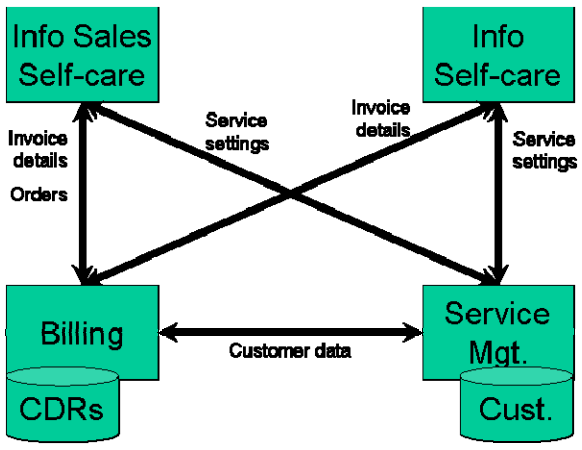

Figure 4 Application architecture

4 Middleware introduction

As the previous chapter described the example environment, this chapter will elaborate on the typical solution when middleware is introduced.

4.1 Eliminating point-to-point interfaces

The most obvious reason to implement middleware is to reduce the interface complexity. Figure 2 showed the example where all individual systems have interface connections with all other systems. Worst case, for N systems, this leads to ((N * (N-1)) / 2) interfaces. Figure 5 shows the typical situation when middleware is implemented and each system only has one interface to the middleware system. So at the expense of introducing a new system (the middleware), the number of interfaces is reduced to N. For N=4 (as in this example) this is an improvement from 6 to 4 interfaces. This seems only a small improvement, but in a typical environment, N=20, leading to an improvement from 190 to only 20 interfaces. And, if implemented well, this is the kind of improvement that justifies the expensive implementation of the new middleware system.

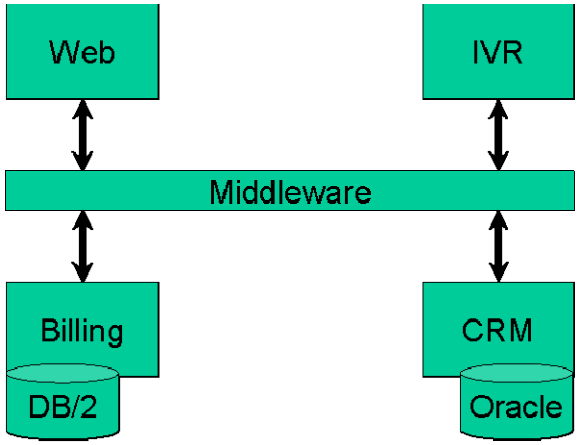

Figure 5 Introduction of middleware

4.2 Infrastructure architecture

In most situations, the improvements can be seen best in the infrastructure architecture of the new situation. Figure 6 shows the infrastructure as designed for the example presented in chapter 3. It shows that all communication is now handled over TCP/IP connections and each system only has to implement a single interface technology to interact with the middleware system. The messy interface details are ‘hidden’ in the adapters which are part of the middleware configuration. There is a dedicated adapter for each type of interface which takes care of the translation of the technology specific message structure to a generic message structure, e.g. from one data structure into an other.

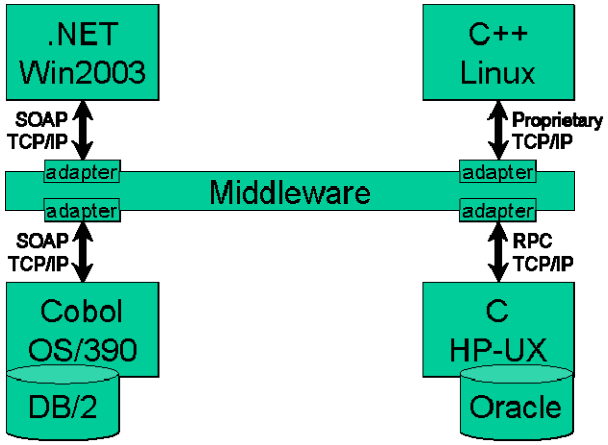

Figure 6 Middleware infrastructure architecture

4.3 Application architecture

In the most ideal situation, the same benefits as introduced for the infrastructure would also apply for the application architecture. Figure 7 shows how this would work out for the example at hand. Both front-end systems send the information requests to the middleware system which propagates each request to the appropriate back-end system. Each back-end system in turn only needs to implement only once the services it needs to provide.

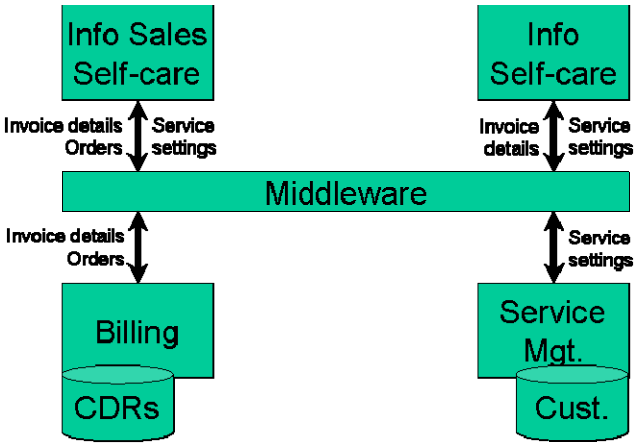

Figure 7 Middleware application architecture

5 Functional pitfalls

As the previous chapters are needed for the introduction of the necessary background information, this chapter will discuss the pitfalls that are the subject of this article. Most pitfalls are introduced because of the direct migration of the existing implementation to the implementation with middleware. On the infrastructural level, the benefits are clear and implemented straight forward as indicated by Figure 6. The pitfalls that are presented here arise on the application level and can be described as follows:

1. Different methods
2. Different functional levels

5.1 Different methods

A typical pitfall occurs with the one-to-one migration from an existing implementation as shown in Figure 8. Although both front-end systems provide the same functionality, the implementation differs on details. In the example, the customer has the option to enable or disable the voice mail function. The Web uses an explicit method setting the value to 'On' or 'Off'. The IVR however uses an implicit method which toggles the current setting to the opposite value.

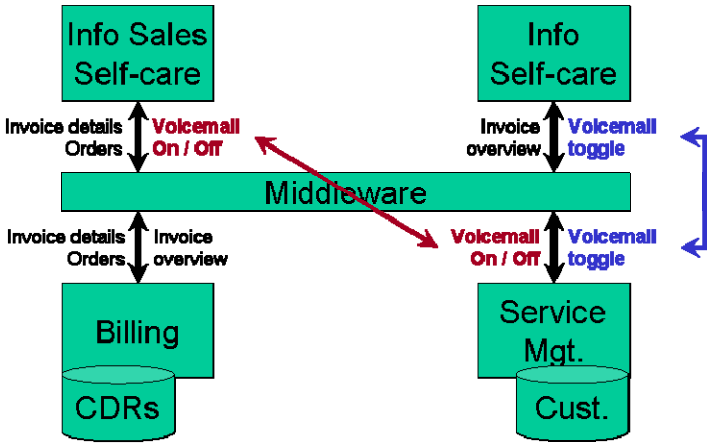

Figure 8 Functional pitfall regarding different methods

Figure 9 shows how this problem can be solved by implementing a single method and update one of the front-ends to support the new method. Although this pitfall seems obvious it occurs often in real environments because of the inherent complexity of the implementation with multiple systems. Also, in most companies, development is organized in teams which are dedicated to a single system, not knowing what the functionality in other systems might be.

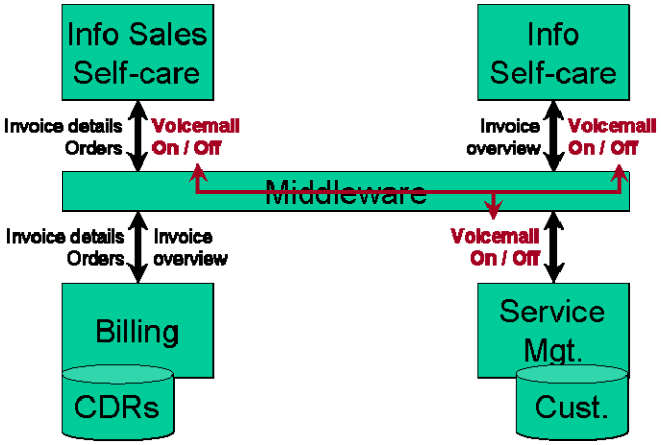

Figure 9 Functional solution regarding different methods

5.2 Different functional levels

Another pitfall is related to different functional levels as indicated in Figure 10. Opposite to the IVR system, the Web can show much more information. This led to the situation where the IVR only presents invoice overviews while the Web presents invoice details. Again the 1-on-1 migration would result in two dedicated messages with different content destined for the different front-ends.

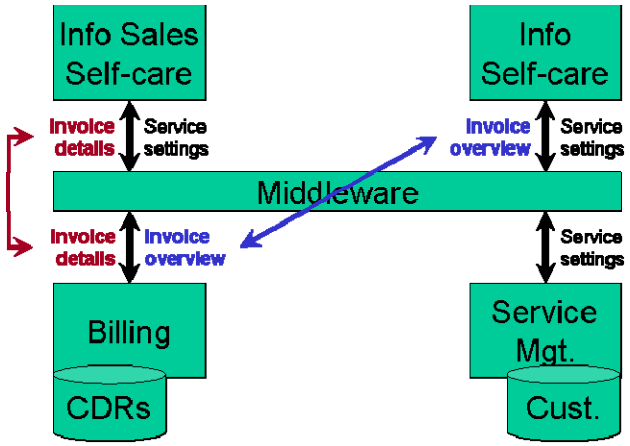

Figure 10 Functional pitfall regarding different functional levels

Figure 11 shows how the different front-ends both use the same functional level, thus sharing the same message structure which contains the most information. In this case, it is the responsibility of the IVR system to filter out the information that is not needed. This filtering can be done by the IVR system itself or by the middleware system. In the latter case, the adapter (as shown in Figure 6) must convert the data rich message into a leaner message which only contains the data needed by the IVR system.

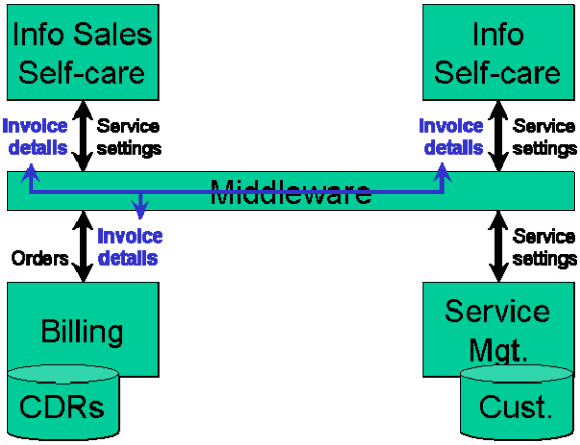

Figure 11 Functional solution regarding different functional levels

6 Conclusion

As indicated in chapter 4, introductions of middleware usually concentrate on the infrastructure architecture. Chapter 5 however showed some pitfalls which arise in the application architecture. And although the examples used seem rather straight forward, reality shows that these pitfalls often occur.

So in order to gain the full benefit of the introduction of middleware, the conclusion is to equally concentrate on both the infrastructure and application architecture. The best solution is to install a team of architects with a scope of the complete application landscape that needs to be covered by the middleware implementation.