1 The Business Perspective

1.1 The Sponsor's View … and a Few Other Opinions

Mr Evan Gelist, freelance technology evangelist and conference speaker:

"Web services enable dynamic e-business and will be deployed everywhere in the near future. They form a new computing model. There have been other distributed computing approaches, but this time it's serious."

Mr Lee Gassi, manager of IT development unit at PremierQuotes Group:

"I can't see what Web services are good for. This is yet another reinvention of the wheel, the most pointless hype in years. There are interface descriptions and an RPC communication style – I hear CORBA and DCE calling. Portable data? We've got EDI. The XML syntax is way too verbose and the HTTP protocol is not reliable at all.

"I'll challenge the performance and security characteristics – two concerns that always work for new technologies. Wait a minute, I just said that this isn't all new. Anyway, the concerns remain.

"At PremierQuotes, we'll stay with our own data exchange formats, because I have full control over what I define myself. I don't believe the evangelists … they say they have standards, I say they have plenty of competing ones, which is worse than having no standard at all. Enough said!!!"

Mrs Penny Kieper, head of the mid-office business unit at PremierQuotes:

"I'm skeptical about the promises Web services are supposed to bring to the enterprise and tired of playing buzzword bingo. I am only willing to invest in something that gives me real added value and certainly will not pay for technologists merely playing with the latest toolset.

"My questions are: What real business pain is solved, where are the usage scenarios? How do these things help me to run my business more efficiently? Are Web services ready for the real world?"

In this kickoff chapter to the book, let us find out whether Evan is right and respond to Lee's and Penny's questions and concerns. The following sections do so:

- Web services business drivers, business benefits and general advantages
- Common usage scenarios for Web services and related functional requirements
- Typical inhibitors to Web services success and our countermeasures

1.2 Web Services – Holy Grail or Déjà Vu?

As the statements made by our three fictitious characters show, the attitude towards Web services can be anything between euphoria and rejection, including uncertainty. Our objective in this chapter is to present our point of view, which we developed during numerous project and presales situations we have been involved with.

Due to the nature of the subject, this chapter will be a rather smooth start into the Web services world, giving you ideas and selling points for potential application areas in your domain. If the material is too lightweight for you, feel free to directly proceed to the Training Perspective – or even the Architecture and the Development Perspective if you are already familiar with the base Web services concepts.

Motivation. There are two fundamentally different ways to approach the topic at hand. In the book title, we promised to provide *Perspectives on Web Services*. Perspectives is a plural form, so let us start both ways:

- Web services can be viewed as the latest, dynamic stage in the e-business evolution, following static Web sites and transactional Web applications designed for human users. You might have heard vendor strategy names such as IBM's *e-business on demand* and *dynamic e-business*, Sun's *smart services* or Microsoft *.NET*.
- Web services can also be viewed as a simple, low cost enterprise application integration vehicle supporting the *cross platform sharing of functions and data*.

The first view is certainly more visionary than the second one. The driving force for this view is that established e-business technologies such as portals targeting human end users do not address all business problems that occur when business is pursued over the Internet.

Most introductions on Web services mainly emphasize these dynamic elements of the technology. In this book, we cover these aspects as well. However, as we will see shortly, the sharing capabilities of Web services offer tremendous benefits even without the dynamic elements. Let us look at such Web services business drivers and benefits now – choosing Web services will pay off in many cases, even if the vision of dynamic e-business never becomes real.[1]

1.2.1 Business Drivers and Benefits

Our observation is that the actual business benefits facilitated by Web services solutions mainly depend on the *business scenario* at hand, as opposed to the long-term vision. More precisely, the business drivers for *Business-to-Business* (B2B) and *Business-to-Consumer* (B2C) scenarios are quite different.

[1] Consequently, challenging the vision is not an excuse for not getting involved.

However, the distinction between B2B and B2C is not yet sufficient to enter the discussion about Web services business drivers and benefits; we have to further distinguish between *Application-to-Application* (A2A) and *Human-to-Application* (H2A) solutions.[2] Figure 1.1 illustrates these four scenario and solution areas as well as the resulting relationships between them:

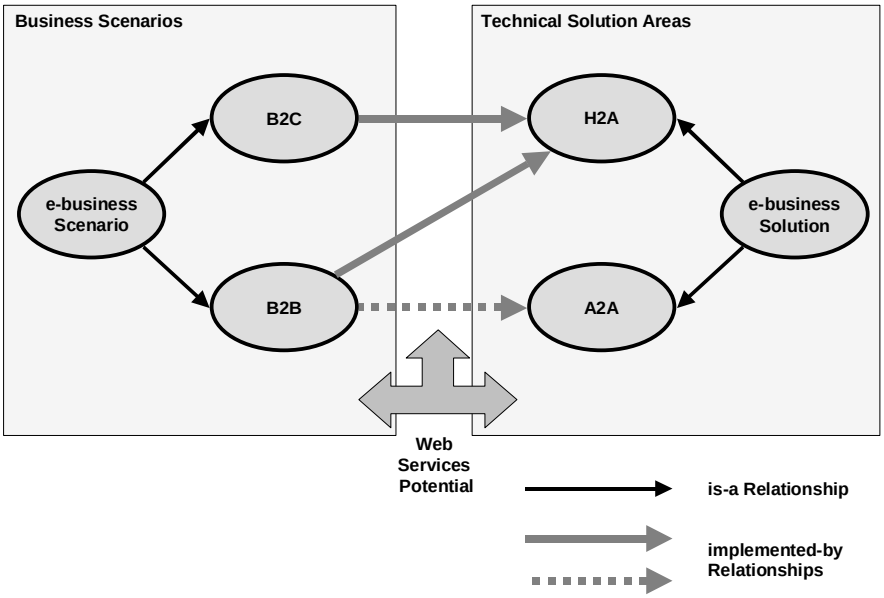

Fig. 1.1 e-business scenarios and solutions

As the figure shows, B2C almost by definition means H2A – consumers use browser clients to access services exposed by server side Web applications via HTML/HTTP. The first generation of e-business solutions, static Web presences, can be viewed as a simple instance of H2A in this picture. Transactional Web applications designed for human users form the second wave of H2A. There is little Web services have to offer to consumer-oriented H2A solutions.

Today, many B2B scenarios are addressed with H2A applications as well. For example, a supplier portal targeted at procurement staff falls into the B2B/H2A category. However, on both sides of the communication link, IT systems are likely to be involved. Hence, the procurement portal shifts the human-to-machine interface, or *media break*, from the supplier to the purchaser. Much more business benefit can be realized with A2A solutions avoiding any media break. In the evolution of e-business, B2B A2A consequently is the next logical step, following the portal trend the industry has gone through in recent years.

[2] We do not introduce these acronyms here for their own sake – we will use them occasionally throughout the book when pointing out differences between Web services and other e-business solutions.

Web services are an ideal platform to implement such A2A solutions for B2B scenarios. Figure 1.2 comes back to our procurement example and outlines the process differences between the traditional (non-IT), the H2A and the Web services A2A approach (the media breaks are indicated by flashes):

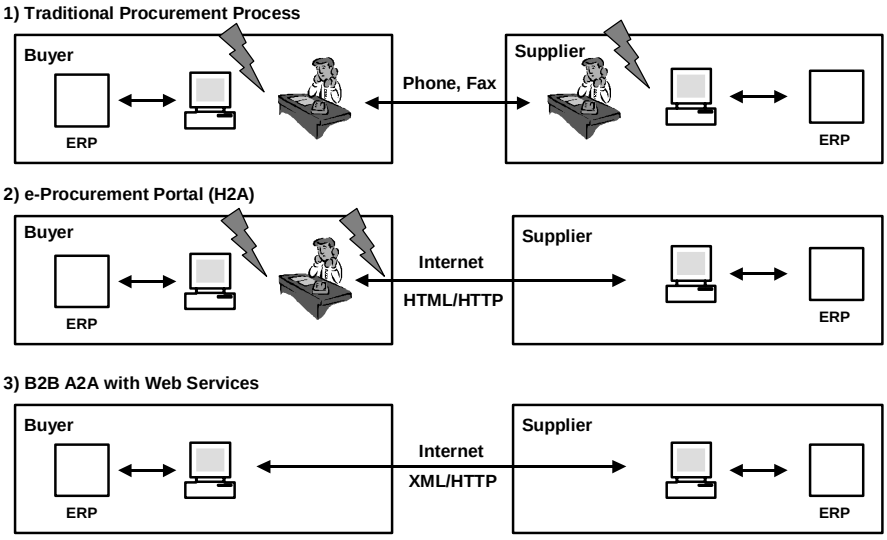

Fig. 1.2 B2B evolution

With Web services, the Enterprise Resource Planning (ERP) systems on procurer and supplier side directly communicate with each other via *eXtensible Markup Language* (XML) [16] document exchange over HTTP – there no longer is a media break. Compared to the H2A case, the data transfer format changes from HTML to XML; the transport protocol, HTTP, remains the same.[3]

1.2.2 Requirements for Application-to-Application Communication

Having outlined the B2B A2A drivers and business benefits, let us now walk through the requirements for such solutions, linking them back to the benefits:

- Automation through application clients
- Connectivity for heterogeneous worlds
- Information and process sharing
- Interface agreements
- Reuse and flexibility
- Dynamic discovery of service interfaces and implementations

[3] HTTP is the *Hypertext Transfer Protocol*, HTML the *Hypertext Markup Language*. These two technologies link human Internet users with websites and Web applications.

- Business process orchestration without programming

Not coincidentally, these requirements are identical to a large extent with the technical characteristics of the Web services technology.[4]

Automation through Application Clients

To support B2B A2A scenarios, arbitrary software applications running in different organizations have to directly communicate with each other. In the supplier portal example, the client side ERP system, not the client staff, should access the supplier portal.

Web services leverage widespread Internet technology to give such *application clients* the ability to talk to remote systems. Thus, human-to-human or human-to-machine interfaces are removed from inter- and intra-company business process execution; the level of automation increases tremendously.

When the transition from H2A to A2A business process automation is made, significant cost reductions can be achieved. For example, manual tasks such as printing, fax communication and data entry into IT systems can become obsolete if Web service interfaces are available.

Connectivity for Heterogeneous Worlds

Web services make it possible to *connect* many different computing platforms. For the first time, a single simple technology is available for very different programming language environments such as Java 2 Enterprise Edition (J2EE), Microsoft .NET, C++, SAP ABAP, Lotus Domino, Perl and many more. All hardware and operating system platforms with HTTP protocol implementations and XML parser support can participate.

XML, which is the foundation for the Web services messaging language *SOAP* [13], is not just a beautified variation of ASCII or EDI revisited. Numerous applications are able to process XML, and the XML notation is defined in public standards – the dream of truly portable data has become reality. Sometimes XML is referred to as the "Esperanto of the computing world".[5]

The underlying business benefit here is investment protection, as all technological camps in a company can continue to work with the technology they are familiar with. Existing skills and assets can be fully leveraged – if you imagined Web services as a baseball cap, the label on the inside would say "one size fits all".

---

[4] Admittedly, we reverse engineered these requirements to some degree – Web services prove that sometimes technology can drive business.

[5] As of today, not too many people speak Esperanto, but that's a different story.

Information and Process Sharing

A universal communication and integration offering such as Web services allows you to *export* and *share* both data and business processes between companies or business units.

Information sharing here means making already existing company-internal data available to business partners and customers. Two simple examples are the exchange of customer profiles and production statistics.

Supply chain management along the vendor chain and synchronization of customer-related data across lines of business are two examples for shared business processes.

Through information and process sharing, the relationships with customers and business partners can be intensified. Moreover, a company can attract additional customers and business partners and improve its organizational efficiency. Business process execution speeds up.

Interface Agreements and Tool Support

As data and process sharing is frequently required, the IT industry has come up with numerous solutions in this area already.[6] Examples include:

- Homegrown string and Comma Separated Value (CSV) formats
- Key-value pair exchange formats stored in Java hash maps or other containers
- The commarea in CICS and the copybook in COBOL
- SAP R/3 IDOC records

A typical element of pain when building such solutions is that there has to be a *formal contract* between client and server before data and functions can be shared. Such a contract is cumbersome and time-consuming to define and agree upon.

Most of the mentioned approaches to process and data sharing have limited built-in interface description capabilities. Development, test and maintenance efforts therefore limit the large scale usability of these approaches.

The Common Request Broker Architecture (CORBA) with its Interface Definition Language (IDL) tackles the problem. However, for various reasons, CORBA so far has failed to become widely accepted. Commercial Enterprise Application Integration (EAI) products also provide solutions, which usually are proprietary or address only parts of the problem.

The Web services technology addresses this issue by making use of the XML schema standard [39] and the *Web Services Description Language (WSDL)* [21]. It is worth noting that formal Web service specifications are simpler to develop and maintain than, for example, EDI definitions.

As we will see in the Development Perspective, to define a Web service interface for an existing function can literally be a matter of minutes if proper tools are available. Therefore, early adopters can initiate projects even if standards bodies

[6] As a matter of fact, the existence of such solutions is one of the reasons why many of the elements you find in the Web services technology look familiar.

have not yet agreed upon common process and data definitions.[7] The standardized interface can quite easily be adopted at later stages (again making use of tools available).

Another advantage of a lightweight formal interface contract is that parameter validation can take place early, for example in the client or middle tier rather than the backend of a multi-tier solution. An example is the check whether a part number passed to a supply chain management system is syntactically correct. Data quality often is an issue for home-grown, character stream-based solutions.

Reuse and Flexibility

Web services do not intend to reinvent the wheel. Existing application components can easily be integrated regardless of implementation details. This flexibility makes the technology ideal for companies operating in a decentralized fashion as well as companies pursuing outsourcing, merger and acquisition initiatives.[8]

Sometimes *relationship without responsibility* is claimed to be a Web services benefit as well. This statement only applies to the technical level, as a client application does not have to make any assumptions on how its communication partner is implemented – architects call this *information hiding*. A client application does not have to know the implementation platform of the server side.

Commercial, legal and logistical responsibilities certainly still exist; no serious business would be possible otherwise. Service provider and consumer have to agree upon mutual contractual obligations and levels of service quality. For example, a reaction on service unavailability must be defined (for instance, a fallback process).

Business benefits that reuse and flexibility help to realize are cost and risk reductions, as well as an improved agility and anticipation of changes.

Dynamic Discovery of Service Interfaces and Implementations

If you want, you can let your application clients dynamically, i.e., at runtime, look for and download both service address, service binding and service interface information. Architects call these features *location transparency* and *late binding*.

According to the vision outlined at the beginning of this section, such services will eventually become available; the technical foundation is there. However, we have only seen experimental stages of such truly dynamic e-business so far. This is not a surprise, as a new business model accompanying the new programming model has to evolve.

Among the business benefits dynamic service discovery brings to the table are increased flexibility (for example, the transport layer connecting client and server can be switched at runtime) and competition between different service providers.

---

[7] We call this approach the *sneaker* principle: “Just do it!”.

[8] Which company is not investigating or already undertaking such efforts these days?

Business Process Orchestration without Programming

The final requirement Web services promise to address is business analysis-driven *orchestration* of smaller units of work, such as sub-processes or activities, into aggregated business processes. It should be possible to execute such orchestrated processes automatically (without requiring human interventions or programming effort, that is).

This vision, which is known under the *workflow* banner,[9] is an area to which Web services bring tremendous advantages. An existing workflow can be made accessible through a Web service interface. Vice versa, the invocation of a Web service can be viewed as an atomic workflow activity. In the past, the workflow technology has been missing this openness to some degree. With Web services, the programming effort is minimized if suitable composition tools and workflow engines are used.[10]

Specifications and the first wave of tool support for Web services workflows are already in place. Process orchestration via Web services is not yet in widespread use, but promises to realize significant business benefits such as shortened development cycles and productivity gains.

Have we already convinced you?

The requirements for A2A communication and the capabilities of Web services match, even if a conservative view is taken. Furthermore, Web services can even drive new business opportunities and increase revenue.

If you have not yet identified where Web services will be of help in your organization, keep reading. We have not yet listed all advantages of the technology, and concrete sample scenarios are yet to come as well.

1.2.3 Additional Advantages

As we have already seen, other approaches addressing the A2A business problems exist. In this section, we therefore want to discuss a few of the unique selling points of Web services.

Non-Invasiveness

Web services have (almost) *zero impact on the existing IT infrastructure*. For example, the client side Web services support comes as a very lightweight *stub*. Communication without any such stub is supported as well.

The required server software, which comprises an XML parser, an HTTP server and a servlet engine (in case the server is implemented in Java), is available al-

[9] You might also have heard the term *Business Process Management (BPM)*.

[10] These tools change the programming model, which might introduce a role conflict between developers and business analysts now being able to define *executable* workflows.

most everywhere.[11] There is no dedicated Web services product that has to be purchased, installed, managed and maintained. The required server components come with the application servers such as IBM WebSphere Application Server, which are widely deployed in many companies as of today.

Productivity Boost and Industry Momentum

As *convenient high level client APIs* can be generated from the formal service interface contracts, Web services increase the programmer productivity tremendously; the time to market for A2A solutions is shortened. Many other integration approaches suffer from the primitiveness of their programming interfaces, which frequently are text-oriented and/or not well documented.

Another key point is the *strong vendor support* Web services receive. For example, code generators and other tool support for Web services are available on the market. Service deployment can also be performed by off-the-shelf tools.

Comparing Web services tooling with a roll-your-own philosophy, product offerings implement the Web services and XML specifications and best practices much better than any home-grown toolset can ever do, as economical constraints apply to on-the-project tool building.[12]

Standardization and Openness

Web services are a *future proof technology* due to the standardization of the underlying specifications. There are *de jure* standards such as W3C recommendations and IEEE Request for Comments (RFCs), as well as *de facto* standards from organizations such as OASIS and Web Services Interoperability (WS-I).

Furthermore, many building blocks are *open source* software. The textual wire format is human readable, which simplifies debugging and interoperability testing. The portability requirement is addressed as well, at least in the Java world, through the Java Community Process (JCP). An important example is the JAX-RPC specification [80], which defines the standard Java SOAP programming interface.

Low Project Risk

Web services are an inexpensive and (almost) *risk free integration and communication* strategy. Project teams can leverage existing experience with Web application development and e-business in general. The technical foundation, XML and HTTP, is well-established in the industry – many companies already use these technologies excessively.

---

[11] We assume that HTTP is used as transport protocol here; other alternatives are available.

[12] Usually it is a good thing if you can put pressure on a vendor rather than your own project team if a tool does not work or any features are missing.

Bottom Line

By now, you hopefully have enough arguments to be able to convince your prospective sponsor to invest in a Web services project.

As a lightweight, universal glue technology is available, technological limitations or concerns regarding implementation effort are no longer excuses for not integrating existing application islands upon demand.

1.2.4 Litmus Test (a.k.a. Applicability Filter)

This subsection helps you to decide whether convincing your sponsor is not only feasible, but also a good idea – many, but not all integration scenarios call for Web service technology usage.

Let us start with two high level filters, which most candidate services will pass:

- Any application that does not have a GUI, but provides a supporting function for more than one other solution component can become a Web service.
- Any proprietary (e.g., text-oriented) data exchange format lacking tool support can be replaced with a standardized Web services interface.

Next, let us investigate the business process to be implemented. The A2A implementation of a business process usually yields a high return on investment if the process is well established and executes frequently, e.g. at least once per day. The automation of the process should make one or more human activities obsolete.

The chosen process should be reasonably well-understood and have documented semantics. In a first-of-a-kind project, the process to be automated should be simple. The elements of project risk, which we will discuss in the Engagement Perspective, should be manageable and not drive you out of business in case they occur.

Qualification Checklist. We are now ready for the business level litmus test (in the Architecture Perspective, we will present a technical counterpart to it). If the answer to any of the following questions is “yes”, you should consider initiating a Web services project for your scenario at hand:

- Do you want to interact more tightly with your business partners or share information with them?
- Is there a requirement to link company internal functions, which currently reside in separate application *stovepipes* (business unit-specific application islands, that is)?
- Are there any legacy business application assets, for instance written in COBOL, that you want to make available for reuse and assembly in additional programming languages?

- Do you have to integrate Commercially Off-The-Shelf (COTS) applications into an already existing application environment?
- Does your IT strategy foresee rich application clients, in response to the observation that browser clients are suited for many, but not all types of applications? Does your network setup not allow you to connect such clients to the server with binary or proprietary protocols, e.g., due to a rigid firewall configuration policy?
- Are you looking for a more flexible overall IT architecture that can immediately adopt to changes in the marketplace?[13]
- Is a best of breed strategy in place?
- Is your system environment heterogeneous?

There are many more application scenarios for Web services one can think of – this technology is very universal. Just be a little creative and you will certainly find examples in your organization. If nothing helps, remember some integration efforts that failed on the *technical* level. Try again with Web services, this time you can get it right.[14]

Before we continue with usage scenarios, let us briefly look at Web services versus J2EE and EAI. This discussion is the last step of our small litmus test.

Web services versus J2EE and classical EAI

There is a place for both Web services and J2EE as well as Web services and traditional EAI approaches.

J2EE is an implementation platform, Web services are a communication and integration technology. It is important, but not sufficient, to standardize the interface descriptions and message exchanges (as Web services do). A standardized development platform such as J2EE is required as well. In most chapters of this book, it is our working assumption that you have decided for J2EE. The decision-making regarding the Web services implementation platform is therefore out of our scope.

As for Web services versus EAI products, it is impossible to come up with a general recommendation when to use which alternative. This could be a quality of service decision, or a matter of personal/company preferences. Coexistence is possible; at the time of writing many EAI product vendors such as IBM Crossworlds were in the process of Web services-enabling their offerings.

[13] Let us play buzzword bingo: *competitiveness*, *agility* and *responsiveness* are required.

[14] We do not claim that this technology solves organizational issues such as company politics as well.

1.3 Usage Scenarios

Having discussed general business drivers, benefits and advantages that Web services bring to your domain and having defined the litmus test, let us now investigate the main business scenarios for which Web services are a valid and promising architecture alternative:

- Enterprise Application Integration (EAI)
- Business-to-Business (B2B)
- Common Services (CS)

We will also provide a collection of miscellaneous scenarios in which Web services provide true value-add for real-world business problems. All presented scenarios pass the qualification checklist we presented in the previous section.

Before diving in, let us first define an A2A complexity matrix.[15] Figure 1.3 shows how we position the three application areas EAI, B2B and CS in the matrix:

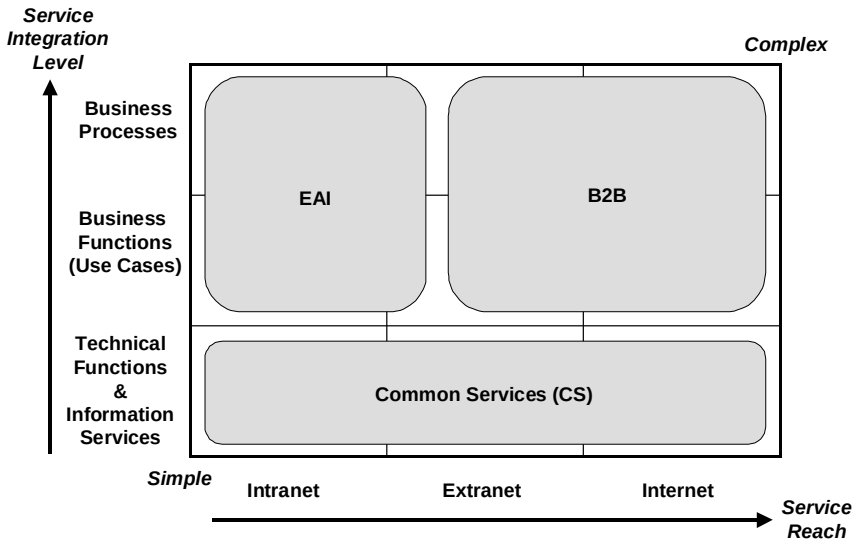

Fig. 1.3 EAI, B2B and Common Services and Web services complexity

Let us now walk through the matrix, starting with the introduction of its vertical and horizontal axes. These two dimensions of this matrix are the *service integration level* and *service reach*.

In the vertical axis, the service integration dimension, the three levels are:

1. Elementary functions

[15] There has to be some high level positioning of this kind in a chapter called *Business Perspective*.

   a. *Information services*, which normally perform read-only access only
   b. *Technical functions* such as basic general-purpose utilities
2. *Business functions* (a.k.a. use cases) with transactional behaviour; transactional behaviour here translates into read-write access to service data.
3. *Business process* externalization along the value chain, which can optionally be defined dynamically.

The following levels of complexity spawn the horizontal axis, the service reach dimension:

1. Company internal services, connecting employees and organisational units via an *intranet*.
2. *Extranet* scenarios, supporting a closed business partner community or an industry-specific user group.
3. Services with global reach, covering the entire *Internet*.

As Figure 1.3 indicates, common services is our generic term for simple technical and informational services; such services can have any reach. EAI and B2B are more complex and therefore challenging endeavours. The main technical difference between EAI and B2B from our perspective is the different reach. This is no hard separation, though; it resides somewhere in the extranet level, as extranets can connect different companies as well as different units of a single business.

What we learn from this diagram is that the internal implementation of a simple information service or technical utility is significantly easier to achieve than a worldwide B2B undertaking.[16]

1.3.1 Enterprise Application Integration (EAI)

Our discussion on EAI is split into a discussion of the requirements and a sample solution scenario.

Requirements

Today any business must be agile, i.e., responsive to market opportunities. Therefore, companies demand more efficient and optimized development cycles and want to be able to easily integrate applications from multiple vendors, possibly running on different platforms.

The ability to reuse and reconfigure existing business components is key to being agile. Furthermore, it has to be possible to leverage existing application assets for additional sales channels and other application areas.

A company internal integration layer has to provide information about, and access to, simple, self-describing components that can be invoked from multiple ap-

[16] This holds true with and without Web services, so this might have become clear without the diagram; however, we found that positioning certain scenarios and solution architectures into this diagram is a good exercise during project initiation.

plication contexts. The integration layer must be independent of any specific implementation language, transport protocol, operating system and hardware platform – very likely, the existing company infrastructure is heterogeneous. The integration layer has to be supported by open standards, development tools and off-the-shelf applications.

The Web services technology meets these overall requirements, as we have discussed when presenting its benefits and general advantages.

EAI-specific challenges. In addition to the general integration layer requirements, certain EAI-specific issues have to be addressed in EAI solutions as well:

- Abstraction level, granularity of programming interfaces and call frequency
- Correlation and matching of entity identifiers, e.g., database keys
- Ownership and synchronization of data – mirroring to be avoided or balanced
- Data transformations, either implemented as runtime conversions in adapters or requiring modifications of the participating applications
- Event-driven processing, as opposed to user interface-centric computing
- Compensation and rollback strategies for transactional integrity

Web services solutions might be able to deal with these EAI issues, depending on the architectural decisions being taken during solution design. We come back to these decisions in the Architecture Perspective.

An EAI Scenario

Figure 1.4 on the following page shows a fictitious EAI scenario, in which a two dimensional region- and business unit-oriented application landscape is the status quo. Depending on the region, different sub-unit structures are in use (sub-region versus business function versus product). The IT systems of the business units are likely to be implemented differently in each region; a true integration nightmare results.

The fictitious company overcomes its application island status quo by introducing an *integration hub*. Through this hub, different business units can exchange data. Furthermore, applications residing in different regions can obtain information from each other. Two cross unit applications in the company headquarters can also use the hub services to retrieve information from the regions.

Such an EAI architecture can help to solve many business problems. The headquarters can obtain an overall view on the business through a single technical interface. Up-to-date information becomes available in a single data mart; business forecasts can therefore be improved. The regions can also synchronize their logistics systems with each other. The same data is used in all business applications – unnecessary and error prone replications are avoided.

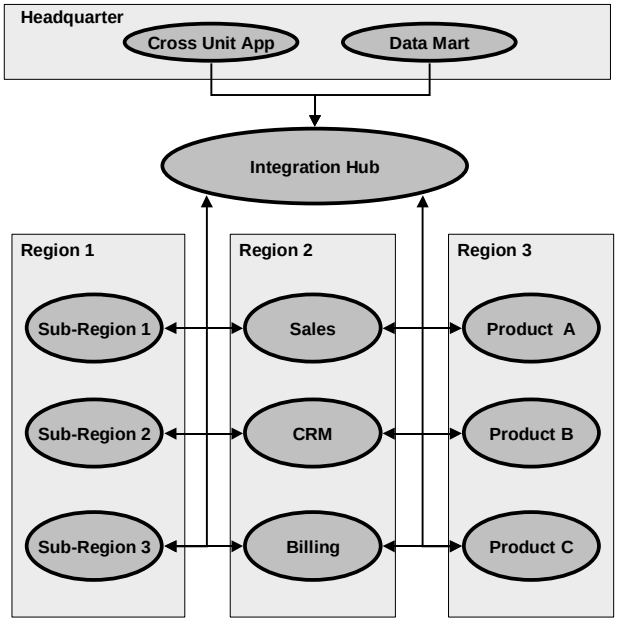

Fig. 1.4 An EAI scenario

Solution outline. While it is quite easy to come up with a diagram like Figure 1.4, it is a true challenge to actually implement such an integration architecture. All participating applications have to connect to the hub. They either must support a common data model, which has turned out to be unrealistic in many EAI projects, or support certain interfaces and protocols, which requires many transformations. We have listed several other challenges in our discussion of EAI requirements above.

For now, we merely want to point out that the benefits and advantages that Web service bring with them, for example flexibility, cross platform capabilities and formal interface contracts, simplify the implementation of such EAI scenarios tremendously. In the Architecture Perspective, we will come back to this scenario and show how Web services technically solve the related integration problems.

1.3.2 Business-to-Business (B2B)

As in the previous section on EAI, we take a look at typical B2B requirements first and define a sample scenario next.

Requirements

Many B2B requirements are shared with company internal EAI issues similar to those stated in Section 1.3.1. B2B in its A2A flavor can therefore technically be

viewed as *external EAI*; classical EAI would then also be called *company internal EAI*. Following this observation, we merely look at B2B specific issues now.

B2B-specific challenges. The objective of B2B is business partner integration via an extranet or the Internet, or, in other words, to make already existing internal data and functions available for business partners and customers.

When exchanging information across enterprise boundaries, message syntax and semantics have to be defined unambiguously and a common interoperable transport channel has to be agreed upon. Entity identifiers such as customer keys, contract numbers and bill of material part numbers must be either unique, so that they can be shared, or mapped on each other.[17] Public and private service flows and interfaces have to be distinguished.

Let us start with the bad news. Web services do not address the semantics problem. Concepts such as UDDI tModels and UUIDs [37], which we will introduce in the Training Perspective, ease the problem; however, Web services do not provide a full solution to it.[18] For business and service identification, complementary solutions are required, for example worldwide company numbering schemes such as the one from Dun & Bradstreet.[19]

The good news is that the message syntax and transport channel to be used are defined in Web service descriptions; these agreements are part of the formal WSDL interface contract. Therefore, B2B interoperability can be achieved more easily with Web services.

A B2B Scenario

Let us take a look at a marketplace centric procurement solution (Figure 1.5 on the following page). Three parties and a total of seven applications participate in this scenario; seven interactions are displayed as well. Note that the example does not aim to be complete and realistic; it merely serves to illustrate the business problem.

Each application in the scenario might have been implemented differently. For example, the sales application might be based on Microsoft Office, the accounting system might be an SAP module and the billing application a home-grown Java solution.

The message syntax and protocol question might also have been answered differently for each interaction in the diagram. For example, XML over HTTP might be in use for marketplace catalog update, EDI might implement order tracking and bill presentment. Other communication technologies might support the remaining interactions.

---

[17] Note that these issues can also apply to company internal EAI; however, in that case, corporate standards might already have been defined.

[18] In the Future Perspective, we will take a look at the semantic Web movement, which promises to do so. Other initiatives such as RosettaNet and ebXML exist as well.

[19] Dun & Bradstreet, http://www.dnb.com, assigns D-U-N-S identifiers to businesses.

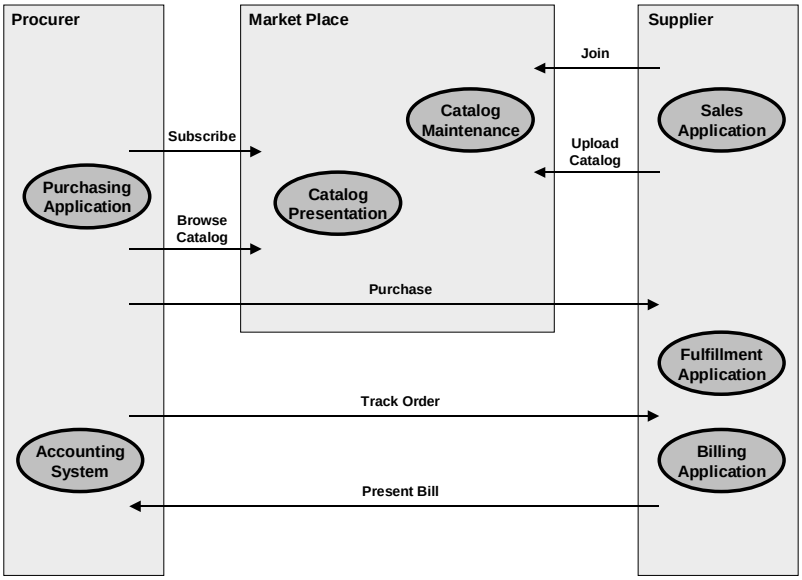

Fig. 1.5 A B2B scenario

Solution outline. In such a situation, several nontrivial implementation projects are required to connect both the procurer and the supplier to the marketplace. The vision of profitable automated trading relationships can hardly be achieved this way. This integration scenario therefore calls for a single message exchange infrastructure for all involved platforms. Web services can provide such a universal communication layer.

In the Architecture Perspective, we will come back to this scenario and show how the Web services support for this scenario can technically be implemented.

1.3.3 Common Services

We define common services as any piece of software which provides a certain functionality that is not specific to any business process. It is desirable to develop such common services only once and use them in different application scenarios.

Common Services Requirements

The motivation for developing common services is to make existing functionality available to many different clients. A few examples for such functionality are:

- Logging and other utility services
- User profile, personalization, collaboration and other typical portal services
- Access management services such as authentication and authorization

As some or all of the software components using such common services might run in different processes, each common service either has to provide a remote invocation API or it is instantiated multiple times. The latter alternative might turn out to be expensive, for example in terms of licensing costs.

A Web service interface to the common service can act as a remote cross platform invocation API: XML is an universal data format and HTTP a common inter-process communication protocol. Note that we do not say that each and every service in a layered software architecture should expose its interface as a Web service; this only makes sense if at runtime some sort of system boundary is crossed and/or there are multiple types of service clients.[20]

The decision whether a certain service should become a Web service or not is an architectural one; we come back to it in the Architecture Perspective, where we provide a detailed checklist as well as related patterns and anti-patterns.

Common Services Scenarios

Here are some more examples where a Web service interface for existing common services can provide universal access to valuable features:

- Company-wide address service and website directory service
- Component (or service or utility) inventory
- Resource and systems management interfaces

On the Intenet, you can find many portals listing numerous examples for common (Web) services. Two such portals are XMethods, www.xmethods.com, and Salcentral, www.salcentral.com.

At the time of writing, these portals featured services calculating the distance between two geographic locations, assigning postal codes to street addresses, sending emails and SMS to mobile phones, providing interfaces to operating system event logs, etc. When you read this book, many additional examples will already have become available on the Internet.

1.3.4 Miscellaneous Scenarios

We have also come across several additional Web services application areas. Note that most of these scenarios actually are more technical than business-related.[21]

[20] Otherwise the overhead might be too high; in such cases, other architecture alternatives such as RMI/IIOP with better performance characteristics are available.

[21] Should such technical information appear in a chapter named *Business Perspective*? Well, the listed technologies characterize the scenarios well. An exercise for you would be to find out where in your company the technologies mentioned in this section are already used and to identify the underlying business scenarios.

Information Retrieval Services

Read only information system access comprises a large group of Web services scenarios. Examples include query interfaces to document management systems, to company-wide reporting systems (data marts) and to search engines such as Google.[22]

Many of the simple Web service examples such as weather report and stock quote retrieval, which are commonly used to introduce the technology, fall into this category.

Structured e-Mail and XML Document Exchange

Homegrown *structured e-mail transfer* and *XML document exchange*, realized via HTTP connections or message queues, can easily be re-implemented with Web services. Key reasons for undertaking such migration efforts include client side API improvements, tool support and reduced deployment and maintenance effort on the server side.

Typical business scenarios in which such data exchanges take place are the order processing in Web applications and semi-automated company internal workflows.

File Transfer

Similarly, *file transfer* realized via the standard File Transfer Protocol (FTP) or other protocols can be replaced with Web services interfaces for document up- and download. Batch-oriented information exchange is still very popular in the industry, and Web services are a standardized, lightweight and flexible alternative to other transfer protocols.

A Web services-based file exchange can be especially useful in scenarios where the recipient has to immediately react when data comes in. With FTP, the receiver would have to poll for incoming data. Another reason for replacing an FTP-based transfer solution with a Web service could be that certain security restrictions only allow HTTP as a transport protocol.

RMI/IIOP Substitute

Rich client to middle tier communication and classical two-tier client-server information exchange is another area where Web services should be considered as an architecture alternative (as a substitute for RMI/IIOP).

If software distribution is an issue, firewall constraints limit the applicability of RMI/IIOP (for example for security reasons) or the use of Enterprise JavaBeans is prohibited in general,[23] Web services *can* be a valid architecture alternative for the rich client scenario.

---

[22] Google provides a Web service interface, see http://www.google.com/apis

[23] Note that you can write J2EE-compliant applications not containing any EJB.

Additional Application Areas

Let us touch upon a few more potential application areas for Web services briefly.

EDI replacement. Web services are a low cost, straightforward-to-pursue alternative to EDI. Other advantages of Web services are their simplicity and transport layer independence. EDI comes with its own data and metadata definition formats, which have to be supported on both sides of the communication link.

Application portal adapters. Content syndication is a key issue in B2B/B2C portals; if all content providers offer a Web service interface, a single simple component in the portal is able to gather the entire portal content. The *Web Services for Remote Portals* specification, which is available from http://www.oasis-open.org/committees/wsrp, addresses this promising application area.

Core competency-focused organizations. Web services encourage and enable the outsourcing of processes which are not core business. For example, a company in the finance industry can hardly differentiate itself from competition by improving its human resources processes or its claim processing, and might therefore want to sub-contract these tasks. This is an orchestration-oriented business model – streamlined organizations emerge, which solely aggregate third party services.

Mobile device to enterprise application communication. A business example is route optimization for a parcel delivery service: an optimization algorithm could run on the company server and transmit an XML representation of the best route to the truck driver's personal digital assistant (PDA) via a Web service invocation.

Peer-to-peer and Grid computing. These two areas are currently emerging. Peer-to-peer initiatives such as JXTA, http://www.jxta.org, currently do not use WSDL and SOAP. In contrast, the Open Grid Services Architecture (OGSA) leverages and enhances Web services as introduced in this book. In the Future Perspective, we will provide a little more insight into such OGSA *Grid services*.

Web services versus enterprise services

Note that WSDL can do more for you than just describing your *Web* services that can be accessed via SOAP over HTTP. For instance, IBM has defined additional service bindings for plain Java, Enterprise JavaBeans and software components providing a Java Messaging Service interface.

Furthermore, the J2EE Connector Architecture and WSDL share many characteristics. IBM has therefore decided to use WSDL descriptions also in its resource adapters to Enterprise Information Systems such as SAP R/3 and CICS. Such services are called *enterprise services*, as Internet protocols do not have to be involved.

In this book, we almost exclusively focus on Web services, which can be reached via SOAP/XML messages transmitted over HTTP or other transport protocols. However, in the Development Perspective, we will briefly visit the IBM WebSphere Studio support for enterprise services.

1.4 Potential Inhibitors

Having identified the business drivers, benefits and most imminent application scenarios for Web services, we now switch to a more skeptical point of view for a moment. In numerous consulting sessions, we have come across the following main inhibitors to a widespread adoption of Web services:

- Over-enthusiastic expectations
- Goal conflicts
- General skepticism regarding maturity of new technology
- Security and performance concerns
- Logistical and organizational issues
- Skill deficiencies
- Roll-your-own temptation

Our objective in this section is to make you aware of these inhibitors and some of the arguments you might hear when presenting the idea for a Web service solution to stakeholders and other involved parties.[24] We also present a few ideas on how to mitigate these issues, as unfortunately most of them cannot be ignored or discussed away if you want your project to start and eventually succeed.

1.4.1 Over-Enthusiastic Expectations

Web services are a very universal technology and make use of already existing assets such as XML and HTTP. The terminology in use is partially borrowed from other distributed computing approaches. Hence, irritation is likely to occur.

Moreover, Web services currently receive a lot of public attention. This information overkill and over-selling might cause unrealistic expectations regarding what can be achieved, particularly on the first project you conduct. Such misperceptions and unrealistic expectations can easily lead to unnecessary disappointment and subsequent rejection of the technology.

We recommend to *let pragmatism rule* in reaction to this phenomenon. You should assist with the separation of hype and reality. In other words, try to clarify what Web services have to offer and which problems they do not address.

For example, when presenting this technology to an audience that has not yet been in touch with it, position the technology correctly; talk about both benefits and downsides, patterns as well as anti-patterns. Work with concrete examples your audience can relate to – our checklists and scenarios will help you to easily identify such examples during your workshops.

You should tailor any analogies you use to the background of your target audience. For example, do not talk about CORBA all the time during a presentation if the session participants have never heard of it.[25]

[24] We have heard these arguments time and time again.

[25] You might guess why we give this advice here. We made this mistake a few times.

1.4.2 Goal Conflicts

Web services are about opening up systems and data to a new type of clients. Such information sharing might be a breakthrough achievement from an overall company perspective.

Nevertheless, chances are not everybody might be supportive, as having exclusive access to certain data means being powerful and influential. Furthermore, there is a tradeoff between openness and predictability; for example, data quality and consistency issues that might have been existing for years might become obvious when new interfaces are added. Finally, it might become more difficult to maintain service levels if a new type of client applications is connected to an existing system.

As a consequence, certain parties might argue against Web services projects opening up systems to client applications residing in other domains. Very likely, some of the other potential inhibitors have to serve as (pseudo) arguments, as usually nobody wants to admit the existence of the goal conflicts.

You cannot handle this problem on the technical level. It is pointless to praise a solution architecture if there are unarticulated concerns (i.e., a "hidden agenda"). An enterprise architecture consulting project might help in the long term. Meanwhile, try to establish an *alliance* of sufficiently influential stakeholders that directly benefit from the Web service solution.

A more detailed discussion of this problem, which at its heart is not Web services-specific, is out of the scope of this book.

1.4.3 Skepticism about New Technology

General resistance to change might articulate in phrases similar to the following ones, which are probably more frequently thought than articulated:

- "e-business, the Web and Java are new, not mature and therefore evil."
- "Anything that does not run on platform XYZ is unreliable."[26]
- "I do not know all these acronyms and therefore I do not believe in them. I stay with what I have been using for the last several years."

Such killer phrases are hard, if not impossible to break. Using standard consulting techniques, such as structured interviews and workshops, you can try to identify and respond to the technical concerns. However, arguments do not help if the real concerns are emotional ones. A solution, or rather workaround, would then be to follow the sneaker principle mentioned earlier and simply go ahead, ignoring the concerns. You could call this approach *Web services by stealth*.[27]

[26] XYZ representing z/OS, UNIX, Win32 or any other platform, depending on the camp the speaker belongs to; this is not the point here.

[27] Support (and funding) from some less skeptical people are still required though.

If your project succeeds, you can convince the skeptical colleagues afterwards. This suggestion assumes that your corporate culture allows such an approach and you are willing to take the risk.[28]

1.4.4 Security and Performance Concerns

Security and performance are two important areas of non-functional requirements. Here they represent any other potential technical gaps Web services, as any other relatively new technology, might be facing.

Security

Explicit security features had originally not been build into the core Web services protocols. This was not an accident, but a conscious decision, in order to keep the first set of specifications simple. Transport mechanisms such as HTTP offer basic capabilities, but not all end-to-end security requirements can be addressed.

The short term solution to this issue is to carefully analyze the actual security requirements such as authentication, authorization, non-repudiation, confidentiality and integrity. You should also investigate how existing H2A browser applications, as well as the existing manual (non-IT) implementation of the business process, are approaching them. In many cases, already existing security technologies such as those used to protect the network and the transport layer provide sufficient functionality to meet the requirements in an acceptable way for the short term.

In the mid to long term, implementations of the emerging *WS-Security* standards will solve the problem. The definition of a *migration story*, for example starting with existing transport layer security features and upgrading to Web services-specific security support later, might remove the security concerns.

Refer to Architecture and Operational Perspective for more technical arguments regarding this topic.

Performance

Acceptable performance is a key requirement for any IT system. Now, what is *acceptable* performance? The client expectations, which typically are not well articulated, have to be met. The performance question can only be answered for a solution, not for an entire technology, as many factors have an impact. Hence, we can only make some general statements here.

According to our experience, performance is not an issue in most scenarios. All general Web application performance engineering guidelines apply. The overhead of the toolkits we have been using in projects is reasonable; XML document

[28] The first step towards mitigating the risk to fail is to continue reading. In this book, we try to guide you around the pitfalls (more precisely, the pitfalls we are aware of).

transmission has a natural overhead.[29] Countermeasures such as XML document *compression* and tag minimization exist. We will discuss some related best practices in the Engagement Perspective.

In general, make sure that realistic values for end-to-end response time and throughput are explicitly stated and written down at the beginning of the project. Take reasonable architectural decisions, select suitable hardware and test early and extensively. Performance then no longer is an area of concern.[30]

We will come back to performance aspects when taking the Architecture and the Operational Perspectives on Web services.

1.4.5 Logistical and Organizational Issues

Many non-technical issues, too many to mention them all, might inhibit the success of Web services. Among the most prevalent ones are unclear roles and responsibilities, lack of hardware and software resources and budget constraints.[31]

Roles and responsibilities simply are a project management issue. Note that the hardware and software for Web services projects can be shared with other Web application development activities.

By the way, Web services are an excellent topic for side and interim activities in projects, for example while the main stream is paused for budget or other reasons. Tremendous progress can be made and demonstrated, while only limited effort has to be spent. For example, it is feasible to schedule one-day workshops, in which you start with the identification of an integration scenario and end with the demonstration of a working solution.

1.4.6 Skill Deficiencies

A common misunderstanding about Web services is that advanced skills such as a deep knowledge of XML are required to leverage this technology. The adoption of Web services in an enterprise might slow down if such skills are not available.

Fortunately, such advanced skills are not really required. The SOAP implementations provide automatic encoding support. Mature development tools are available to help as well.

It is therefore sufficient to be able to read XML and understand the namespace concept. In other words, you should have the skills taught in the first few pages of each section in the Training Perspective.

---

[29] Web services are never going to match the capabilities of low-level information exchanges. However, in most cases the benefits are large enough to afford the overhead.

[30] A standard project management approach in this area is to investigate and address any remaining concerns early in the project (e.g., through small proof-of-concept activities).

[31] Budget cuts are not an uncommon phenomenon a couple of years after the “dot com crash”, with many country economies struggling.

1.4.7 Roll-Your-Own (RYO) Temptation

This problem is often also called the *not invented here* syndrome. Framework builders and other development teams tend not to trust any other code than their own, hence there might be a lobby preferring to transfer native XML (or even plain text) messages over HTTP.[32]

The approach to convince the RYO camp can be *competition*: initiate a small Web services proof of concept, competing against an existing implementation. Implementing a Web services demo for two to three project-specific use cases usually is a matter of hours or days. Normally you can convince your target audience with this productivity boost made possible by the Web services tooling. When presenting the Web services prototype, use arguments such as API usability and compare the programming tasks for the team with and without Web services tooling.

Note that if somebody does not want to be convinced, it is always possible for him/her to come up with a justification for not doing something (just as it would be possible to argue for doing something). Therefore, even if your arguments are good, you still might not be able no convince the RYO camp, for example due to the emotional involvement with a homegrown solution. Look for other allies in this case.

1.4.8 So Do the Inhibitors Really Inhibit Us?

Technically, Web services are ready to be used in real-world projects delivering enterprise scale solutions. Some tactics might be required for buy-in and just as with any other innovative approach, you will not be able to convince everybody. As long as you convince the right people, this might not hurt too much.

Bottom line

Even without being extremely visionary, there are many usage scenarios and benefits for Web services. Inhibitors such as over-optimism, resistance to change and skill deficiencies can be overcome if you are aware of them.

[32] We have even seen teams implementing a complete SOAP stack without using any standard runtime or tool. Such an approach is a good way to build up skills and experience in proof-of-concepts and technology adoption type of projects. However, it does not scale and is certainly not appropriate for full scope projects that want to deliver a production solution within time and on budget. The long term maintenance effort is another anti-RYO argument.

1.5 Introduction to the Case Study

Throughout the remainder of this book, we will be following a sample company which builds its first Web services solution. This section serves as an introduction to the business drivers for the Web services initiative. In subsequent chapters, we will then present the component model for the Web service solution, explain the technologies selected to build the sample and actually implement and deploy this solution.

1.5.1 Background Information

Our sample application starts with a fictitious insurance company, PremierQuotes Inc., which specializes in the high end of the home insurance market. After a number of successful years selling comprehensive policies, PremierQuotes has saturated the market and desperately needs to find new ways to increase its revenue without diluting the PremierQuotes brand image.

Only a few miles away is the small office which houses fictitious DirtCheap Insurance Ltd. The company was started on a shoestring, hence its low operating costs and targeted marketing have enabled it to quickly become one of the major players in budget home insurance.

In order to fulfill the growth expectations of its stakeholders, PremierQuotes acquires DirtCheap Insurance with a hostile takeover and forms the PremierQuotes Group.

1.5.2 The Business Problem

One year after the merger, the two organizations' policy management systems are still operating independently of each other. PremierQuotes has a number of concerns about the high number of claims being made by DirtCheap Insurance policyholders and its concentration of customers in certain areas. It therefore decides to implement two new innovative mid-office applications:

- A *risk management application* which enables it to have a single view of the total insured value of its policyholders and claims made, broken down by major postal code. In the future, the PremierQuotes Group plans to use these statistics to determine premium weightings for both organizations.
- A central *fraud management application* which searches across both policy management systems for a newly-insured address to identify if previous claims had been made against that property.

This situation is a variation of our general EAI scenario (Figure 1.4.) Without a unified integration technology, both risk and fraud management each have to implement two interfaces; one to access the PremierQuotes system, one for Dirt-

Cheap Insurance. Such a peer-to-peer approach would cause significant design and implementation effort.

1.5.3 Solution Outline

At this point, the Web services technology enters the game. Recently having attended a Web services summit keynoted by Evan Gelist, Penny Kieper, the project sponsor at PremierQuotes, is convinced that Web services are well suited as an integration technology to connect the existing IT systems and the new mid-office applications.

A unified Web services interface to both backend systems is envisioned. The benefit of such an interface is that the claim and policy information can easily be retrieved by the mid-office, even if the interface is implemented on two different technical bases.

This scenario meets several of the qualification criteria stated in Section 1.2.4, as a system boundary is involved and existing application assets require an additional access channel. The business benefit of the mid-office applications is the integrated view on the two separate policy management systems.

However, there is a strong lobby at PremierQuotes pointing out that Web services are emerging technology and have not yet demonstrated to be mature enough for a strategically important project such as the development of the new mid-office systems. In particular, Lee Gassi, the owner of the two existing backend applications, is an initial member of this lobby. He also is concerned about the effort for Web services-enabling the existing systems and transaction load.

Penny and Lee agree to schedule a small series of awareness sessions involving Evan as an external consultant. The sessions are accompanied by a two-day hands-on proof-of-concept. After this effort, it becomes clear that implementing such a solution is feasible and reasonable. Implementation effort and project risk are estimated to be quite low, as the existing J2EE assets can be fully leveraged. Stability and performance of the proof-of-concept solution as demonstrated to the executive management of the PremierQuotes Group are excellent.

Hence, according to Penny's and Lee's recommendations, a decision to go ahead is made. The existing system interfaces will be described in WSDL and SOAP will be the communication protocol between the new mid-office and the existing systems.[33] In case more business units have to be accessed from the mid-office applications in a dynamic fashion, a UDDI registry could be set up in a second step.

The Web service solution does not only meet PremierQuotes immediate integration needs. As a side effect, PremierQuotes is now also ready for future mergers and acquisitions. For example, in case of a merger with another company, the mid-office applications can remain unchanged; merely a SOAP adapter to an additional backend system is required. Parts of the system could even be outsourced easily, as all interfaces are formally described in WSDL.

[33] More details on the technical architecture will be discussed in the following Perspectives.

1.6 Summary

Before taking the next steps, let us summarize the key messages of this chapter, which we hope you will carry into your organization.

1.6.1 Key Messages

We talked about advantages and benefits, scenarios for Web services as well as some inhibitors in this chapter.

Motivation. Web services are for applications what browsers are for humans, they allow them to surf the World Wide Web (WWW). Web services simplify application-to-application communication significantly. Existing systems can be opened up and extend their reach to the Internet, intranets and extranets. B2B, EAI and common services scenarios are particularly well supported.

The conservative view is that Web services are a low cost, low risk communication and system integration technology. The required initial investments are low; almost any server side component can easily be Web services-enabled. In the mid to long term, Web services might even introduce a dynamic programming and, consequently, business model.

Advantages and benefits. The Web services technology is about sharing. XML is a universally exchangeable data format. The Web services technology provides the next level, as functionality from simple components to entire business processes can be shared.[34]

Well-accepted Internet standards such as HTTP and XML are leveraged. Web services are implementation platform agnostic and can therefore provide true interoperability and build a bridge, for example between the Microsoft .NET and the J2EE world. All technological camps in a company, such as database specialists, Java programmers and other developers, workflow experts and XML gurus can join … if they are willing to.

Web services reduce complexity by encapsulation and late binding. Service consumers just need to know the service interfaces; backend implementation details are hidden. Web services provide the ability to act as wrappers to legacy applications, the usage of XML is transparent to end user and programmer. Encoding support comes out of the box.

Dynamic interactions and aggregation to higher level business functions are supported. Web services enable just-in-time integration and support dynamic or semi-dynamic communication scenarios. Service interfaces can be discovered at build-time or at runtime.

[34] With the emerging Grid initiative, even computing resources become shareable, as we will see in the Future Perspective.

Analogy: Web services provide the same level of abstraction and convenience for systems communication and integration as third generation programming languages do for algorithms and relational databases do for data management.

Even if there is an overhead, very few people code business applications in assembler languages these days and ISAM access to flat files certainly is less popular than relational databases.[35]

Scenarios. Regarding scenarios for Web service usage, two dimensions can be distinguished, the service integration level and the service reach. Less complex usage scenarios are technical functions and information services (we summarized them as common services). EAI and B2B on business function and business process level are the more advanced scenarios.

We provided a scenario qualification checklist and gave several examples where Web services fit. Table 1.1 takes a final look at the presented scenarios:

Table 1.1 Web services dimensions and scenarios

Scenario	Reach	Integration Level	Example
EAI	intranet	medium to high	Cross region, cross unit information sharing and headquarters integration
EAI	extranet	medium to high	Business process management
B2B	extranet	medium to high	Supply chain information sharing
B2B	Internet	high	Marketplace connectivity
CS	intranet	low	Logging and other utilities Portal services Access management services Address and website directory Component inventory Resource and systems management
CS	extranet, Internet	low	Information retrieval Structured email, XML document exchange and file transfer RMI/IIOP substitute
other	any	miscellaneous	EDI replacement Application portal adapters Core competency-focussed organizations Mobile device to enterprise application communication Peer-to-peer and Grid computing

[35] Some people (have to) code on lower convenience layers – the developers of high-level tools and APIs, for example. You might also have to do so if your non-functional requirements are exceptionally challenging. And do not forget the RYO camp we talked about when presenting the inhibitors.

Inhibitors. In an attempt to view some of the projects and consulting sessions we have participated in through the rear view mirror, we also discussed potential inhibitors to a widespread use of Web services.

Over-enthusiasm, skepticism, resistance to change, goal conflicts, skill deficiencies and roll-your-own temptation are the main inhibitors we identified during projects and consulting sessions. If you are aware of these inhibitors, you can either countermeasure or work around them, as we have seen in Section 1.4.

Return on Investment and Total Cost of Ownership

You might be wondering why we have not talked much about *Return on Investment* (ROI) and *Total Cost of Ownership* (TCO), two measurements not to be missed in a chapter named Business Perspective.

Formal ROI and TCO calculations are a tough business. Concrete numbers depend on many factors and assumptions and are generally hard to predict. Actual ROI and TCO computations have to be out of scope for this book. However, we did talk about cost reductions and mentioned the RYO camp on occasion. A standards-based, tool-supported approach certainly scores better in terms of ROI and TCO than any RYO strategy.[36]

1.6.2 Where to Find More Information

The following resources provide insight on the business aspects of Web services:

- The Component Based Development and Integration (CBDI) Forum, http://www.cbdiforum.com, has published many excellent articles on service-oriented architectures and Web services.
- Analysts and researchers such as ZapThink, http://www.zapthink.com, Gartner http://www.gartner.com and Giga Information Group, http://www.gigaweb.com, have published many reports on the topic at hand.
- The IBM jStart team provides many real-world case studies on its website: http://www.ibm.com/software/e-business/jstart

1.6.3 What's Next

Now that we and the PremierQuotes staff are convinced that Web services are exactly what the IT infrastructure of our company is looking for, how do we get there? As a next step, let us switch roles and drill down one level, taking an XML and Web services course in our Training Perspective. After that, we view Web services from the Architecture Perspective and implement a full-blown scenario in the Development Perspective. Service rollout is covered in the Operational Perspective; the Engagement Perspective discusses project steps and best practices.

[36] This was the last set of buzzwords, at least for this chapter.