2 — Electronic Services

Text

Summary

This chapter provides an overview of two streams – the automation of activities and programming paradigms – which drove and supported the evolution of electronic services. These services are service systems implemented using important elements from the fields of automation and programming. It presents several classifications to understand the nature of services according to different views. Since services can take various forms, the chapter explains and contrasts electronic services, web services, cloud services, the internet of services, and service-oriented architectures.

Learning Objectives

1. Describe the evolution of electronic services as a result of the automation of economic activities and self service.
2. Describe the evolution of electronic services as the improvement of programming paradigms.
3. Classify service systems based on the role of information technology, service architectures, strategies, and business models.
4. Compare the various service types and paradigms that exist: electronic services, web services, cloud services, and internet of services.

Jorge Cardoso and Hansjörg Fromm

Opening Case Amazon Web Services ■

AMAZON EMBRACES SOAP AND REST WEB SERVICES

`Amazon.com` is known by most people as the largest e-commerce web site which sells books and other goods. Less known is the fact that Amazon has close to 1.000.000 servers. Why so many computers? To support millions of customers, Amazon had to build a massive storage and computing infrastructure with high availability and failure resistance to operate its web store. Once the infrastructure was built, it became clear that it could be exploited as a commodity by selling services which would be billed based on their usage. The platform was officially launched in 2006 and named Amazon Web Services (AWS). Nowadays, AWS is responsible for providing services to well-known web companies such as Foursquare, AirB&B, Netflix, Pinterest, Reddit, and Spotify.

What Are Web Services?

The technology selected to make the infrastructure remotely available to customers was web services running over web protocols. Web services are a method of communication between computers using the world wide web. They offer functionalities that developers can use in their software applications to invoke and execute remote functions or methods using open and standardized protocols.

AWS offers more than 40 proprietary web services ranging from e-mail services to sophisticated database services. For example, Amazon SimpleDB provides a highly available and flexible non-relational data store. Amazon Glacier is an extremely low-cost storage service that provides secure and durable data archiving and backup. Amazon Simple Email Service is a scalable and cost-effective email-sending service for the cloud and Amazon Flexible Payments Service facilitates the digital transfer of money.

Hundreds of thousands of companies are using AWS to build their businesses. One interesting example is `Foursquare.com`.

Foursquare Use Case

`Foursquare.com` is a location-based social app used by 40 million people worldwide to check-in to places (e.g., restaurants and stores), to exchange travel tips, and to share locations with friends. The platform performs business intelligence and analytics over more than 4.5 billion check-ins each day.

The major asset of Foursquare is the large amount of data generated by check-ins. The data needs to be continuously stored and processed with business intelligence applications to create reports and long-term trend analysis. In the past, the use of property databases to process data came with high annual licensing costs and the need for qualified engineers to administer the platform.

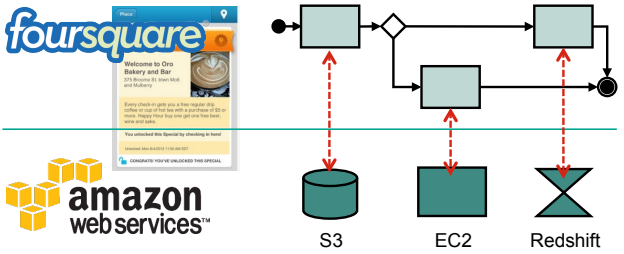


Figure 2.1: Foursquare relies on Amazon Web Services for storage, processing, and business intelligence

To reduce costs, Foursquare uses Amazon Web Services (Figure 2.1). For analytics, it adopted Amazon Redshift, a simple, fast, and cost-effective petabyte-scale data warehouse service which offers an efficient solution to analyze data. Simple Storage Service (S3) was also used to store images and Elastic Compute Cloud (EC2) was contracted for a fast and scalable processing.

The Benefits of Using Web Services

For most adopters, the key benefits of using web services include cost reductions, flexibility, and higher productivity.

Costs The services provided by AWS can free businesses from high initial capital costs. Samsung has achieved reliability and performance objectives at a lower cost by relying on cloud services instead of using on-premise data centers which have high hardware and maintenance expenses.

Elasticity AWS provides elasticity which enables to increase the computer resources allocated (e.g., the number of CPU or the storage available). This elasticity has enabled SEGA to reduce costs by more than 50% with new servers when unplanned load spikes occur after the launch of new games.

Investments Cloud services free organizations from setting up dedicated IT teams and infrastructures. NASA was able to construct a robust, scalable web infrastructure in a few weeks instead of months to support the Mars exploration program (`mars.jpl.nasa.gov`).

But Amazon AWS is not the only player in the cloud computing arena. Amazon is leading the cloud race, but Windows Azure and Google Cloud are not far behind. Other major players such as IBM, Oracle, and HP are also taking a dominant position in this market.

Opening Case ■

2.1 Perspectives on Services

Already in 1968, Fuchs [1] presented a study that clearly indicated that the future employment growth would be almost entirely absorbed by service industries. Nowadays, the most developed countries have become service-based economies in terms of the distribution of people employed. While the first services were certainly delivered by humans to humans, the advances in computer systems over the past sixty years allowed computers to deliver services to humans. *Information and communication technology* (ICT) has significantly contributed to the evolution of services. Over the years, each wave of innovation has created solutions to automatically execute activities that were once done by human beings.

Services and self services were covered in Chapter 1. It shed some light on the difficulties of finding one single definition for the concept of service. One of the causes of these difficulties is related to the fact that the evolution of services can be observed from two distinct perspectives:

- As the automation of economic activities and self service.
- As the improvement of a programming paradigm.

Figure 2.2 illustrates these two perspectives on service evolution. Their most relevant milestones (represented with a circle) will be discussed in this chapter.

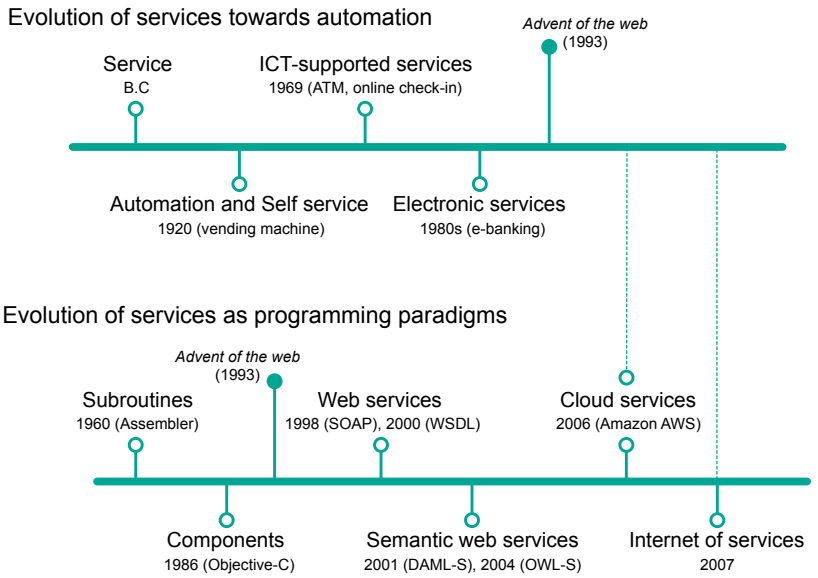


Figure 2.2: The two perspectives on service evolution

The first perspective looks into how traditional services, such as banking and

trading, have benefited from automation and information technologies over the years to enable new forms of service delivery such as e-banking and e-commerce. The second perspective analyzes how the notion of service has emerged in the discipline of computer science, more precisely from the fields of software development and distributed systems, to ultimately give rise to software services such as web services.

A third perspective can also be identified as being the intersection of the first two perspectives (shown with dotted lines in the figure). Cloud services and the internet of services label a set of solutions which have a component of business from the first perspective and a component of technology from the second perspective.

2.2 Services as the Automation of Activities

This section presents the evolution of services in the last 100 years that has seen milestones like the advent of automation and self services in the 1920s, the emergence of ICT-supported services in the 1970s, and the dissemination of the internet which started in the 1990s and has led to a wide proliferation of electronic services.

2.2.1 Towards Automation and Self service

In the old times, most services were delivered by human resources. As the costs of labor increased in industrialized countries, companies were seeking for new ways to deliver services with less human involvement. The answer was automation.

Automation

The manufacturing industry had already introduced automation in their production processes in the early 20^{th} century. At the same time, the first modern (mechanical) vending machines were introduced for stamps, postcards, tickets, cigarettes, chewing gum, or candy. The vending machines automate the sales process, so that no sales personnel are required anymore. In telephony, modern switching systems were invented to allow the caller to directly dial and automatically get routed to the desired telephone number. This was previously only possible with the assistance of a telephone operator. After automation, hardly any operator assistance was required.

Self service

All of these service automation examples have in common that the person delivering the service was replaced by a machine and the customer took over the role of the traditional service person. This was the beginning of self-service as a direct consequence of service automation. Even if not all customers accepted self services from the beginning, the advantages of self services were soon being recognized [2]: The machines – other than shops or offices with service personnel – have no opening hours, they are typically available 24 hours a day and 7 days a week. When using self service, the customer has the service encounter completely under his or her own control.

The customer does not feel rushed, influenced, or pressured by a service person. The customer can easily change his or her mind and interrupt the service process without annoying the service provider. The emergence of electronics in the second half of the 20th century, and subsequently, the increasing use of computers and ICT, accelerated the progress of service automation. Self-service gasoline stations appeared in the 1950s, automatic teller machines (ATM) in the 1960s, interactive kiosks in the 1970s, electronic ticket machines (railway, metro, tram) in the 1980s.

R In 1969, the first ATM made its public appearance by providing a cash service to customers of Chemical Bank in Rockville Center, New York.

Most of the service providers offering self-service technology give their customer a choice. The customers can use the self service, or still request a service person. At a bank, the customer can go to the teller or use an ATM. At many gas stations, the customer can pump the gas by him or herself, or ask an attendant to do it. At the railway station, the customer can go to the ticket counter or use a ticket vending machine.

2.2.2 The Role of Technology

Froehle and Roth [3] have identified 5 different classes that describe the role of technology in the service encounter (Figure 2.3). Each class groups interactions which can be further classified as *face-to-face* or *face-to-screen*.

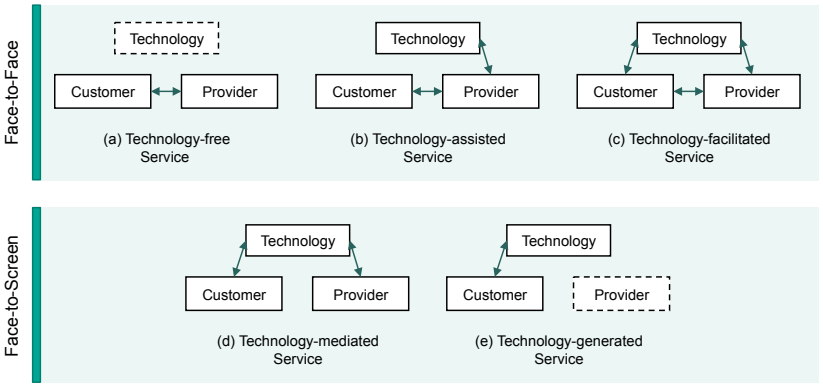


Figure 2.3: The role of technology on electronic services (adapted from [3])

Technology-free and -assisted Services

Technology-free services (Figure 2.3.a) are hardly to be found anymore. Almost every little business like a hairdresser, newspaper store, or beauty salon has nowadays at least an electronic cash register or credit card reader to support their operations.

This would qualify as a *technology-assisted service* (Figure 2.3.b), where the service provider uses technology to improve the customer service, but the customer does not have access to this technology. Other examples are bank tellers, airline representatives, rental car clerks who all have access to the customer's data on their computer terminals while they interact with the customer at the counter. In a technology-assisted service situation, the provider is still in physical contact with or in proximity to the customer.

Technology-facilitated Services

In a *technology-facilitated service* (Figure 2.3.c), both service provider and customer have access to the technology, and are still co-located. An example is an architect who develops a house design interactively with the customer at the computer terminal or a consultant who uses a personal computer attached to a projector to give a presentation to his clients.

Technology-mediated Services

In the *technology-mediated service* (Figure 2.3.d), customer and service provider are not in face-to-face contact. The service is still delivered by people, who, due to the wide dissemination of communication technology like telephone or internet, can work from almost anywhere in the world. Examples are call-center services (communication typically by phone, but also e-mail), internet crowdsourcing platforms such as Amazon Mechanical Turk, and labor-intensive administrative services (accounting, human resources management) provided to companies typically from low-wage countries over electronic networks (also known as *offshoring*). Remote maintenance is an example of a technology-mediated service where a person on the provider's side (technician) directly acts with the belongings of a customer (machines, computers) over a distance.

Example The Amazon Mechanical Turk (`www.mturk.com`), an online service, enables human intelligence tasks to be completed by anonymous people. This is an example of a technology-mediated service, where service providers and customer, not physically co-located, are brought "together".

Technology-generated Services

In *technology-generated services* (Figure 2.3.e), the task of a human service provider is completely replaced by technology. The customer uses the service in a self-service mode. Technology can be mechanical (vending machine), a mixture of mechanical and electronic devices (ATM, check-in kiosk), or purely electronic (home banking). In the context of services, the terms "electronical" or "electronic" are less related with transistors, diodes, or integrated circuits - they mainly refer to the use of information and communication technology. If ICT is the major technology in support of services, one could speak of ICT-mediated or ICT-generated services according to Froehle and Roth's classification.

Technology	Type	Cost
Web	self service	$0.24
IVR	self service	$0.45
E-mail	assisted	$5
Chat	assisted	$7
Phone	assisted	$5.5

Table 2.1: The costs of self services and assisted services

Technology-generated services are gaining a wide acceptance amount companies and governmental agencies for three main reasons. First, self services are generally cost-reducers. In many cases, a customer self-service interaction is approximately 1/20th of the cost of a telephone call. A Yankee Group research report[1] indicates that, within self-service channels, the web-based self-service costs just $0.24 for an interaction and $0.45 for an interactive voice response (IVR), as opposed to $5.50 for a customer service representative assisted interaction via the telephone (Table 2.1). Second, self service can improve customer experience by providing a broad spectrum of choices. Nowadays, the number of features available to e-banking customers has never been so wide. Customers can trade stocks, download bank statements, and make bill payments. Some features may even only be available using a self-service mode. Third, self service can provide a solution to eliminate inefficient provider-customer interaction bottlenecks, i.e., customers having to wait on the telephone for the availability of staff or to wait in a line to be served by customer support. Customers often prefer to perform routine tasks over the internet without requiring any interaction with a representative of an enterprise.

2.2.3 Electronic Services

Both technology-mediated and technology-generated services are better known as *electronic services* or *e-services*. They are labeled as face-to-screen services. The technology required for electronic services can typically be distinguished into three architectural components:

1. Electronic components on the customer's side.
2. Electronic components on the provider's side.
3. An electronic communication line or network connecting the two.

Figure 2.4 gives four examples of electronic services.

[1] `http://www.ccng.com/files/public/Yankee_SelfService.pdf`

Electronic Components

Figure 2.4.a shows an automatic teller machine. In this case, the bank has set up (and typically owns) a user access device (the ATM) and connects it invisibly for the customer with a proprietary communication line to its backend information systems. ATM enabled banks to reduce costs by decreasing the need for tellers.

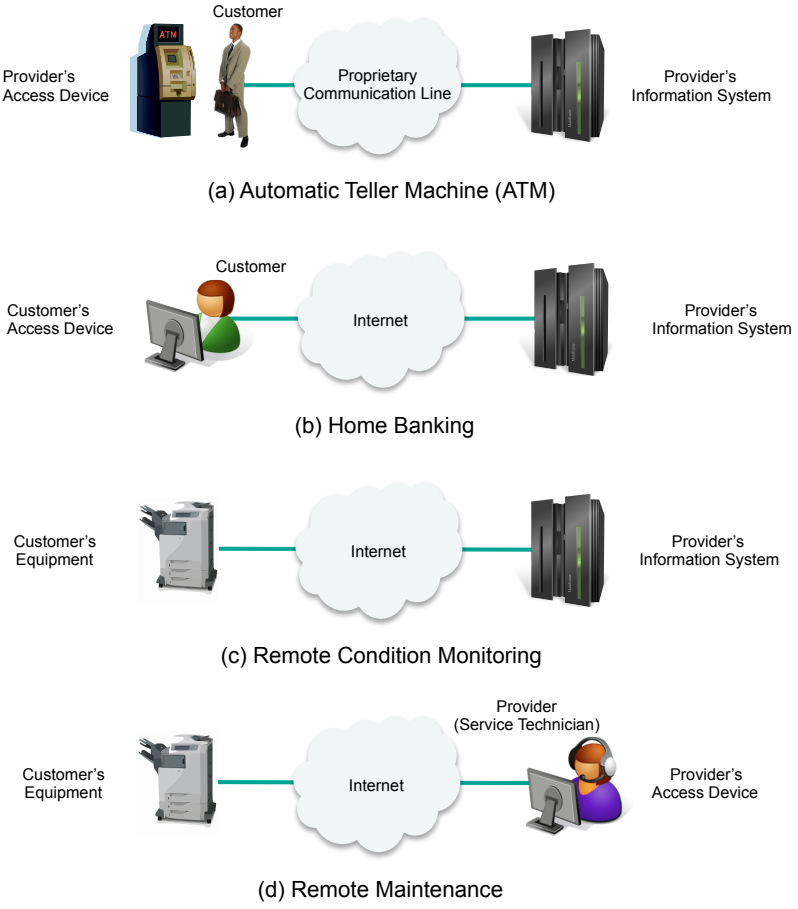


Figure 2.4: Examples of electronic services

Even with the high cost of IT in the late 70s and early 80s, the cost of automated processing with ATM was less than the cost of hiring and training a teller to carry out the same activity. In this case, a complex and specialized machine, interconnected to the bank mainframe, was developed to replace a human activity.

Figure 2.4.b illustrates *home banking*. In this case, the customer has his or her

own access device (which could be a personal computer, tablet computer, or a smart phone) and uses the internet to connect to a computer application that the bank has made available for conducting different financial transactions. Figure 2.4.c and d show examples of *remote maintenance*. In this case, an automated system (c) or a maintenance expert (d) on the provider's side uses electronic equipment to connect to a remote technical system (machine, computer) on the customer's side via the internet to conduct a maintenance task. A person may or may not be involved on the customer's side.

Definition — Electronic Service. An electronic service is a service system (with elements, a structure, a behavior, and a purpose) for which the implementation of many of its elements and behavior is done using automation and programming techniques.

Communication Technology

Common to all of these examples is the use of communication technology establishing an *online connection* between the two sides. For this reason, electronic services or e-services are also called *online services*. The electronic components on the customer's and on the supplier's side can vary. As Figure 2.4 shows, the electronic component on the customer's side might be a device belonging to the provider (e.g., an ATM) operated by the customer, a device belonging to the customer and used by the customer (e.g., a PC or smart phone), or a device belonging to the customer acted upon by the provider (e.g., remote maintenance). The electronic component on the provider's side might be the provider's entire computer center running a number of internet-enabled software applications (for ATM, online banking, condition monitoring) or a personal device (e.g., a PC or tablet) with the appropriate programs that allows the service technician to do remote maintenance. The examples of Figure 2.4.a, b, and c are technology-generated services (no or little human involvement in delivery) and the example of Figure 2.4.d is an technology-mediated service according to Froehle and Roth [3].

In most definitions of electronic services so far, the *mediation role* of ICT received more attention than the *generation role*. Rust and Kannan [4] defined electronic services as "provisioning of services over electronic networks", Rowley [5] as "deeds, efforts or performances whose delivery is mediated by information technology". For the EU [6], electronic services are "services delivered over the internet or an electronic network". All these definitions emphasize the communication component of electronic services and leave the question open, how services are created or generated. Hofacker et al. [7] give a clue in remarking that an e-service is created "through a process that is stored as an algorithm and typically implemented by [...] software". They describe the case of an ICT-generated service in which the service is rendered ("generated") by software programs running on the provider's computer systems.

The First Electronic Services

ICT-supported services have been around since the 1960s. The online information service CompuServ started in 1969, electronic funds transfer (SWIFT) in 1977. Online banking began in 1981. This was long before the advent of the internet. In the early times, telephone networks with modems for data communication were used, and a variety of communication standards. With the advent of the internet, "other electronic networks" lost more and more importance. Nowadays, electronic services are predominantly provided over the internet. And accordingly, most people today equate electronic services with internet-based services. This perception has become particularly common for the short form *e-services*, from which specializations like e-banking, e-learning, and e-government have been derived.

Definition — e-Banking, e-Learning, e-Government. The Oxford Dictionary defines *e-Banking* as "a method of banking in which the customer conducts transactions electronically via the internet", and e-Learning as "learning conducted via electronic media, typically on the internet". The OECD [8] defines e-Government as "the use of information and communication technologies, and particularly the internet, as a tool to achieve better government", and the IRS defines their e-services as "a suite of web-based tools that allow tax professionals and payers to complete certain transactions online with the IRS".

Since electronic internet-based services are offered through web sites and require user interaction via web pages, one might be inclined to call these services *web services*. However, the term web service is already used by computer scientists with a different, precise technical meaning. It is therefore not recommended to equate the technical term web services with the term electronic services in the sense of economic interactions [9] (web services will be discussed in Section 2.3).

2.2.4 The Value of Electronic Services

Many automated self-service technologies like ATMs or ticket vending machines have made services available independent from office hours. The user, however, has to be physically present at the machine in order to request and receive the service. The internet has revolutionized the access to services. Given the widespread availability of the internet, the user needs only a personal device like a desktop, a tablet computer or a smart phone in order to get access to services. Electronic services have inherently several characteristics such as facilitated accessibility and personalization.

Accessibility

With the easy *accessibility* over standard internet devices, the comfort in using electronic services has been brought a big step forward. E-services can be used from home, from public places like libraries or internet cafés, or even while traveling

– given a connection of the access device to the internet can be established using local area networks or wireless communication. Electronic mail or e-mail was the first service widely accepted by the internet community. It has largely replaced traditional paper mail like postcards and letters. And it has the advantage that it is almost instantaneously delivered.

Delivery

The *delivery* of electronic services, though, is not unlimited. For example, they cannot replace an ATM for cash withdrawal. However, they have begun to substitute ticket vending machines. Railway companies, theater offices, and concert agencies allow their customers to buy tickets electronically and print them out on their home printer. Airlines offer the possibility to electronically check-in to flights and print out the boarding pass on paper or receive an electronic boarding pass on the smart phone which can be scanned at the airport. Even if cash withdrawal is only possible at the bank counter or at an ATM, electronic banking has received widespread acceptance for personal bank transactions like paying bills, transferring funds, or viewing recent transactions and account balances. Books, music, and videos can be ordered electronically - and even be delivered electronically, if the customer prefers an electronic version of the media.

Standing in line at authorities and filling out paper forms is an annoyance for many people. Moreover, authorities' office hours are typically colliding with most citizens' working hours. Therefore, it was very important for local authorities and government agencies to offer public e-services to their citizens independent of their limited office hours. Such services can be requesting the issuance of a new passport or renewal of the old passport, reporting changes of address or marital status, or renewing a vehicle registration.

All these electronic or online services are known under terms like electronic mail, e-mail, electronic banking, e-banking or online banking, e-government, public e-services, electronic ordering, and e-ordering.

Human Touch

Traditional human-based services are characterized by the personal service encounter involving *human touch* and *service experience*. Very often, the provider and customer know each other well from past service experiences. The provider quickly understands what the customer wants and can deliver a very individual, personalized service. This advantage was lost in automated services: the vending machine treats all customers the same. The machine is typically programmed in a way that all customers have to follow the same path in order to get the desired result. Even if a customer uses a vending machine the 100^{th} time, the machine would not behave differently.

Service experience is influenced by the outcomes of the interactions that occur between service systems and their customers. The outcomes can take the form of functional, behavioral, and emotional effects and directly shape service experience.

The interactions occur between people, technology, resources, and customers. They are considered to be a crucial part of a memorable service experience.

Personalization

This disadvantage can be overcome with electronic services. Electronic services – even if delivered by technology - provide possibilities to focus on the customer almost like traditional human-based services do. The keywords are *personalization* and *customization* (yielding individuality), *interaction* (leading to relationship), and *localization* (allowing location-based services). In all cases, a difference can only be made if customers are willing to share their personal information like preferences, transaction details, or whereabouts with the service provider. The customer can benefit considerably from personalization and customization, but will accept the service only if privacy and security are guaranteed. For the service provider, gathering customer data can help to provide focused service offerings, enhance the services, and build a profitable customer relationship [4]. More information on personalization and customization is provided by Vesanen [10], on interaction by Bolton and Saxena-Iyer [11], and on localization by Junglas and Watson [12].

2.2.5 E-service Strategy

A notable classification has been given by Hofacker et al. [7] to evaluate the strategic position of services. It includes three types:

1. Complements to existing offline services and goods.
2. Substitutes for existing offline services.
3. New core services.

Complementary Services

Many companies are adding value to their existing offline services or goods by providing *complementary e-services*. Examples are parcel service companies such as DHL, FedEx, and UPS that allow their customers to track parcel deliveries online; electronic companies who offer their customers technical support via their website and provide materials like instruction manuals downloadable on the internet; and retailers who offer mobile apps that enable consumers to scan products and obtain information on country of origin, processing dates, quality, and nutritional value.

Substitute Services

Companies who offer *substitutes e-services* to existing offline services are well known. Amazon was the pioneer in offering online book ordering and thus became a painful competitor of brick-and-mortar bookstores. Spotify, a provider of digital music services, and Netflix, a provider of digital video services, are challenging traditional CD and video stores and video rental businesses. Online ordering offers big advantages over the traditional business: while brick-and-mortar stores are pretty limited in the number of products they keep on stock, this is less the case

for online companies. Electronic products require little space (only computer storage, a single copy), and physical products sold online can be stored in large remote warehouses. This allows the electronic businesses to serve the long tail [13]: this means a much larger variety including niche products. Google Maps can substitute traditional paper road maps. Online news and online stock reports are competing with financial newspapers. Skype, an internet-based communication service, becomes a challenge for traditional landline or mobile telephony.

New Core Services

New core e-services that arose with the dissemination of the internet are search engines like Google or Yahoo, social networking services like Facebook, LinkedIn, or Twitter, videosharing services like Youtube, Vimeo, or Dailymotion, image and video hosting services like Flickr or Instagram, data storage services like Dropbox, or deal-making services like Groupon.

2.2.6 Electronic Business Models

Section 2.2.3 distinguished three architectural components of an e-service: customers' and providers' electronic components, and the network connecting the two. Electronic business models characterize the role that the provider and the customer take during interactions.

Electronic services have enabled the generalization and rapid dissemination of several business models for electronic commerce: B2B, B2C, and G2C. These terms are short forms for Business-to-Business, Business-to-Consumer, and Government-to-Citizen, respectively. While many more models exist, such as G2G (Government-to-Government), G2E (Government-to-Employee), G2B (Government-to-Business), B2G (Business-to-Government), G2C (Government-to-Citizen), C2G (Citizen-to-Government), this section covers B2B, B2C, and G2C models.

Business-to-Business

Business-to-Business refers to transactions between two parties where the buyer and seller are both businesses. Therefore, products or services are not sold to end users. Buyers purchase goods in large quantities to satisfy the demand of their consumers. Two examples are `salesforce.com` AppExchange, a provider of business apps, and `ariba.com`, an e-procurement platform. In B2B environments, the integration of businesses at the application level (known as application-to-application (A2A) integration) was important to improve not only efficiency but also to reduce transaction costs. A2A integration aims to make independently designed software applications work together.

Example Ariba Inc. uses the internet to enable businesses to facilitate and improve their procurement processes. It was founded in 1996. Until then, typical procurement processes were labor intensive and often costly for large

corporations. Ariba supplied the software to create and host vendors' catalogs online as well as the software to enable purchasing staff to remotely buy items. This required to provide electronic services to both provides and customers. In 2012, SAP AG, the largest maker of enterprise applications software, agreed to buy Ariba for $4.3 billion.

Business-to-Consumer

Business-to-Consumer is distinct from B2B since the business (the B) offers products or services to consumers (the C) rather than to other businesses. `Amazon.com` is a good example of a successful business that has initially started its operations using the B2C model. Customers could simply use a browser to interact with the bookstore. Another example is `eBay.com`. Although, eBay started as an auction site involving only end users which would take the role of suppliers and consumers, nowadays, it also enables businesses to market their products to end users. B2C environments involve human-to-application (H2A) or application-to-human (A2H) interactions and communications. The machine-like nature of many activities executed by B2C platforms made it possible to develop services that no longer required a human provider in the loop unless there was a problem.

Government-to-Citizen

Governments are important stakeholders which are recognizing the value of using electronic services for improving citizens experience and lowering costs. Therefore, a third model also emerged: *Government-to-Citizen* (G2C). The goal is to provide one-stop, online access to governmental information and services to citizens, quickly, and easily. This model can offer a large spectrum of services such as employment offers, business opportunities, voting information, tax filing, license registration, and payment of fines.

2.2.7 Government-to-Citizen Services

Despite the high availability and adoption of e-government services in Europe, satisfaction can be improved since it is not yet at the levels of e-banking and e-commerce. Europe aims to increase the use of e-government services to 50% of citizens and 80% of businesses by 2015. While public services such as income tax declaration have already been modernized there is still room for improvements. Recent budget constraints are pushing governments to drastically increase productivity and reduce costs. Electronic services are a targeted area.

The European Commission is addressing these concerns through its 2020 Strategy and its Digital Agenda flagship. The "Digiytal by Default or by Detour?" report published in May 2013 surveyed 28.000 internet users across 27 European countries. The study found that 73% of citizens declare taxes, 57% change their address, and 56% apply for a higher education online. Among the many points focused, five important key findings are highlighted:

Innovation Transformation is needed to drive the development of innovative e-government services. New solutions that use technology in a smarter way, capitalize on social media and collaborative platforms, implement stronger citizen-oriented strategies, and open up data to exploit the value of hidden knowledge are indispensable. Nonetheless, generating innovative services is far from trivial.

Design Governments have not yet fully embraced a service-oriented thinking. A shift needs to take place to change the ad hoc management of services to their systematic design centered around user needs. There is a variation in the satisfaction of online service users, from 41% to 73%, which may be a symptom of the use of different approaches for service design.

Efficiency The increasing number of automated services and self services currently available generate precious data which can benefit from advanced algorithms and analytical tools. Service analytics can extract information to enable public agencies to improve service delivery efficiency and customer experience. The report indicates that 80% of online public services save citizens time, 76% provide flexibility, and 62% reduce costs.

Transparency Currently, services are presented as black-boxes since citizens are unaware of how processes are internally conducted. In fact, the transparency of service execution rates below 50%. This indicates that more transparency is still needed. Governments are providing basic data about their agencies, but information on services that empower citizens.

Business models Opening up data will enable to implement new business models to reach economic gains from various perspectives. The direct impact of Open Data in Europe was estimated at € 32 billion in 2010, with an estimated annual growth rate of 7%. Nonetheless, few governments are exploiting the full economic benefits of Open Data.

There is no doubt that digital technologies will be a fundamental pillar of future public service delivery taking a different direction from how government operates to date. While Europe is taking a serious strategy to implement digital services[2] at the government level, similar efforts are visible at the global scale.

Three of the most advanced governments in this field include the UK, New Zealand, and US. UK adopted a digital by default approach and offers a guide and resources (`gov.uk/service-manual`) for delivering digital services in the government. New Zealand has followed the same steps and is mirroring UK progresses (see `beta.govt.nz`). In the US, The White House issued the directive "Building a 21st Century Digital Government"[3] on May 23, 2012, announcing a strategy to enable a more efficient digital service delivery by requiring agencies

[2]The term digital services has the same meaning as electronic services

[3]The White House, Building a 21st Century Digital Government, May 23, 2012, `http://www.whitehouse.gov/sites/default/files/uploads/2012digital_mem_rel.pdf`

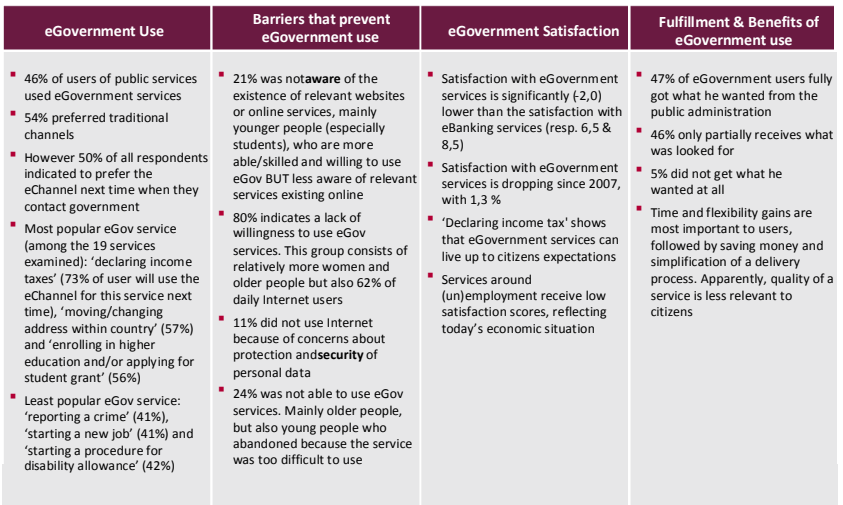


Figure 2.5: Key findings from the "Digital by Default or by Detour?" report [14]

to establish specific, measurable goals by using web performance analytics and customer satisfaction measurement tools on all `.gov` web sites.

2.2.8 Developing Electronic Services

Electronic services are realized by software programs that utilize existing technical infrastructures like the customer's access device, the internet, and the provider's data center and information systems. To develop an electronic service means to develop a software program.

In software engineering, web-accessible programs are called *web applications* (see Figure 2.6). A web application is a software program residing on the provider's *web application server* which uses the client's web browser as the presentation layer in a client/server fashion. Client and server communicate via standard internet protocols such as HTTP. On request of the client, the web application creates a response in form of *web pages* containing static and dynamic content. Programs may be downloaded and executed on the client's web browser. The *database server* will be responsible for the persistence of data stored for the web application.

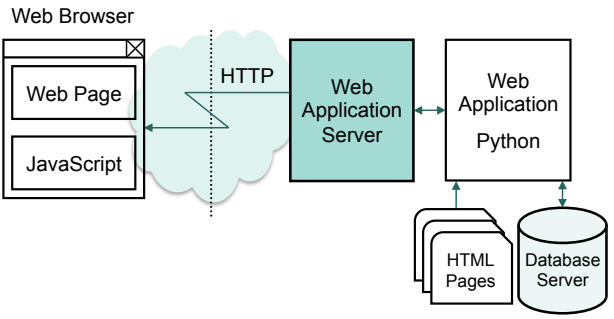


Figure 2.6: Architecture of a web application

Popular programming languages used on the server side of the web application are Java, Perl, PHP, Python, Ruby, or .NET. Most web applications use HTML or XHTML to create web pages. They often send JavaScript and/or Flash to make the content dynamic and more interactive.

2.3 Services as Programming Paradigms

In contrast to the previous section, this section describes an evolution in software engineering that has started with service-oriented programming and has seen a similar boost with the advent of the internet like electronic services. It has led to the concepts of *web services*, *cloud services*, *semantic web services*, and the *internet of services*, which are covered in the following sections.

Naturally, these developments are closer to the fields of computer science and information technology rather than to the fields of economics, business, or management as it was the case of Section 2.2 on services as the automation of activities.

2.3.1 Subroutines

The origins of service orientation can be traced back to the early days of programming. In the 1950s, programmers realized that certain functions, e.g., date conversion routines, were used over and over again throughout their applications [15]. They isolated the function from the rest of the code and put it into a subroutine (see Figure 2.7.a). This subroutine could then be called from any point in the main program where it was needed. It returned the result back to the main program at the point from where it was called. The programmer kept the subroutine in a library for the case that it could be reused in another application. Subroutines could also be shared with other programmers. A sound use of subroutines often reduced the cost and increased the quality and reliability of developing and maintaining large applications.

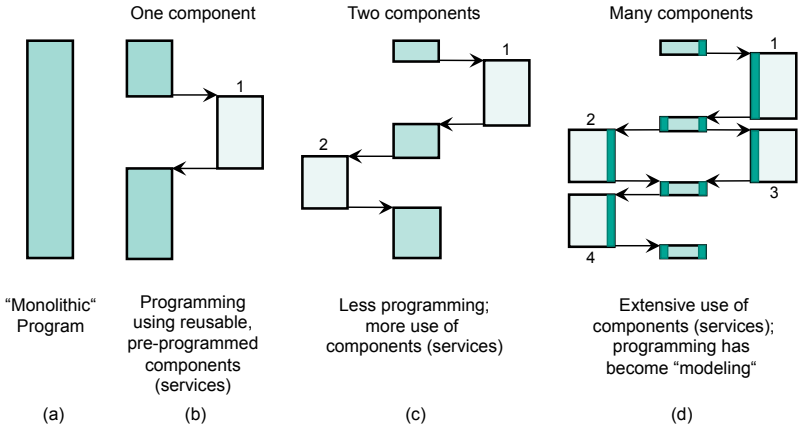


Figure 2.7: Evolution of programming models

2.3.2 Components

Over time, programmers realized that business applications exhibited many of these recurring, separable functions. It became common practice to extensively reuse pre-programmed objects or components as the subroutines were now called (see Figure 2.7.b, c, and d). In the end, programmers did not write many lines of codes anymore – they mainly composed pre-built, reusable components stored in software libraries. The organization of the code in separable functions made code easily reusable due to its inherent modularity, high cohesion, and separation of concerns. This new style of programming was called component-based software development. This style was an indication that components would eventually evolve into distributed services to achieve a higher reuse and decoupling.

Parnas, an early pioneer from the field of software engineering, proposed the concept of *module* [16], which latter evolved into the concept of component. Modules were introduced to enable software developers to implement segments of code independently. They enabled to change one module without affecting other modules, and ease the understanding of the overall system by analyzing one module at a time.

There are some analogies with the French restaurant example introduced in Chapter 1 (page 9): the chef with his staff and kitchen equipment was separated from the aristocrat's household and moved to someplace else, now called a restaurant – much like a particular function was separated from the main application program and moved into a subroutine (component). The restaurant has become a self-contained, autonomous service which (theoretically) could be used by the former owner, but is now available to many other people having a proper physical interface. The subroutine (component) is a self-contained, autonomous program

which can be called from the originating main program, but also (potentially) from many other programs.

An important prerequisite for reusability is the separation of the service's implementation from its interface. Application builders should not be concerned with *how* a service works, only with *what* the service accomplishes and *how* it is invoked. For this purpose, the service needs a well-defined interface and its functionality has to be described in a form understandable by the requestor.

2.3.3 Business Process Modeling

Once a reusable set of services is available, the developer can start building a business application based on these building blocks. By intention, the word programming is no longer used for this activity, but rather the words composing or modeling. This different wording indicates a paradigm shift in software engineering. A widely used approach for the composition of services into business applications is *business process modeling* (BPM). BPM provides graphical notations, which are intuitive to business users yet formal enough to represent complex business semantics.

Business process modeling is used by business analysts and consultants to describe business processes in reengineering projects, and it is used by software developers to document business processes as a starting point for application development. The business process model notation (BPMN) has emerged as a widely accepted standard for BPM. Other well-known graphical languages include event-driven process chains (EPC), simple process state diagrams, and even Petri nets (generally used in academic environments).

Once processes are modeled, they can be executed using an orchestration language. For example, the business process execution language (BPEL) defines processes that can be executed on an orchestration engine. During execution, the externally observable interactions between services is called the choreography and describes collaborations between services.

2.3.4 Service-Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) and Service-Oriented Computing (SOC) are programming paradigms that were introduced to overcome the inflexibility of monolithic software. They utilize services as fundamental elements for developing applications [17].

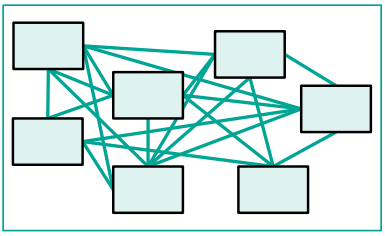
Definition — Service-Oriented Architecture. An architectural style and business-centric programming paradigm to develop distributed systems where systems consist of software clients, which act as service consumers, and software providers, which act as service providers.

Composition and Decomposition

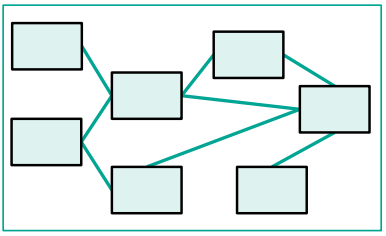
Programmers made the observation that smaller functions like "check customer status", "check order status", or "determine product availability" were repeatedly used in larger business applications like order processing, account management, or service scheduling. It appeared meaningful to separate the repetitive tasks from the business applications and to put them as services into a *service repository*. Once a service repository is available, applications could be developed by *composition* of existing services.

Many companies decided to use the SOA paradigm for new application development. They also reengineered existing applications by breaking them down into smaller, manageable services, which were then glued together again in a more flexible way. This process of *decomposition* is not trivial. There are better and worse decompositions. A good decomposition is one which has little interaction between the components and a high functional coherence within the components. In this case, components are more "self-contained" and "loosely coupled" – characteristics that are desired from a service. In his seminal work "The Architecture of Complexity", Simon [18] points out that many natural (e.g., biological) and man-made systems (e.g., technical and organizational) exhibit this kind of "near decomposability".

Figure 2.9 shows a "bad decomposition" with too many interactions (interfaces) between the components and a "good decomposition" with little interaction between the components. In the latter case, the interfaces are better manageable and can be described with less effort.



(a) A bad decomposition: too many interactions between the components



(b) A good decomposition: little interaction between the components; interfaces can be better managed and easily be described

Figure 2.8: Decomposition of a system into smaller, manageable components

Software Paradigm

SOA has increased the speed of development and the ability to quickly adapt existing applications to a changing business environment. Accordingly, many companies readily adopted the paradigm. SOA is not directly related with standards

and technologies (even if certain standards become prevalent for SOA). Many companies have started to use SOA within the boundaries of their own organization. They used proprietary standards and technologies to implement a service-oriented architecture.

Enterprise Application Integration

Before the introduction of SOA, *enterprise application integration* (EAI) was the traditional solution to integrate the array of systems and applications of an organization. EAI became a top priority in many enterprises as a result of the emergence of the internet and B2B (Business-to-Business). Both accelerated the need for integration. As shown in Figure 2.9.a, EAI was an approach based on a point-to-point integration of systems which created a high complexity in large organizations. The SOA approach enabled to decompose large applications into standard services which were managed by business processes exposing the business logic of organizations (Figure 2.9.b).

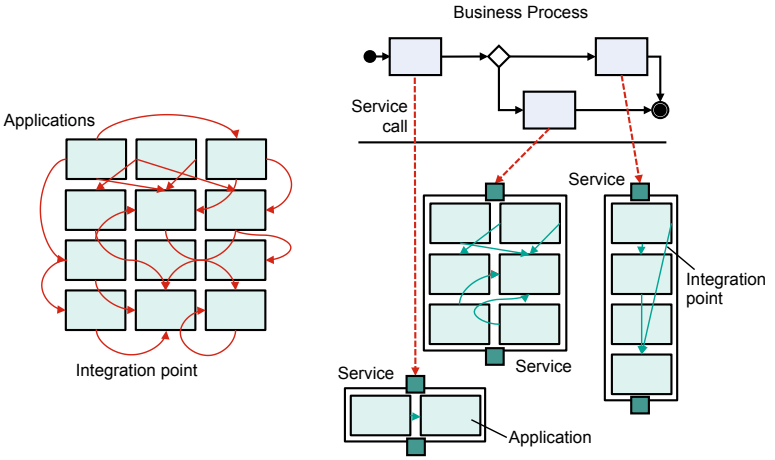


Figure 2.9: Integration of software systems using EAI and SOA approaches

Example — EAI Complexity. The application of an EAI approach requires a systematic and organized management. Otherwise, the number of point-to-point connections can easily reach unmanageable proportions. For example, the grid of applications shown in Figure 2.9.a has 12 systems. Assuming that each application needs to communicate with all the other applications using point-to-point connections, this requires to establish $\frac{n(n-1)}{2}$ connections, with

$n = 12$. Therefore, fully connecting the grid requires $\frac{12 \times 11}{2} = 66$ point-to-point connections.

2.3.5 Web Services

The internet established connectivity between a huge number of people, companies, and organizations. In consequence, software exchange and reuse became theoretically possible between very different and distant software providers and consumers. Such software services provided over the internet are called *web services*. Web services can be seen as an extension of the principles of service-orientation and SOA to the internet.

R In contrast to electronic services, web services are designed for machine-to-machine interaction and have only a programmatic interface. Electronic services can be used by virtually everyone, web services can only be used by programmers.

Web Services and Electronic Services

Web services are often not precisely distinguished from web-enabled electronic services (e-services). Both electronic services and web services are services available over the internet (the "web"). But as it was pointed out earlier, an electronic service, such as online banking, has a user interface designed for *human interaction* (web pages displayed in a web browser). Web services are designed for *machine-to-machine interaction* and have only a programmatic interface. Electronic services can be used by virtually everyone, web services can only be used by programmers. An electronic service can, but does necessarily not need to be developed using web service technologies.

Definition — Web service. The World Wide Web Consortium defines a web service as "a software system designed to support interoperable machine-to-machine interaction over a network".

Service Directory

Prerequisites for the development of web services were mechanisms that allowed providers (programmers) to describe and publish their services, and consumers (application builders) to search and find suitable services (Figure 2.10). For this purpose, a new role has emerged, the *web services broker*, who keeps a *web service directory* listing all web services that service providers have developed, described, and published. The web service directory is often compared with the yellow pages in a telephone directory that allow someone to find a suitable service provider.

Two alternatives of identifying a web service are conceivable:

(a) The web service directory is human-readable and the application builder (a person) looks it up to find a suitable web service.

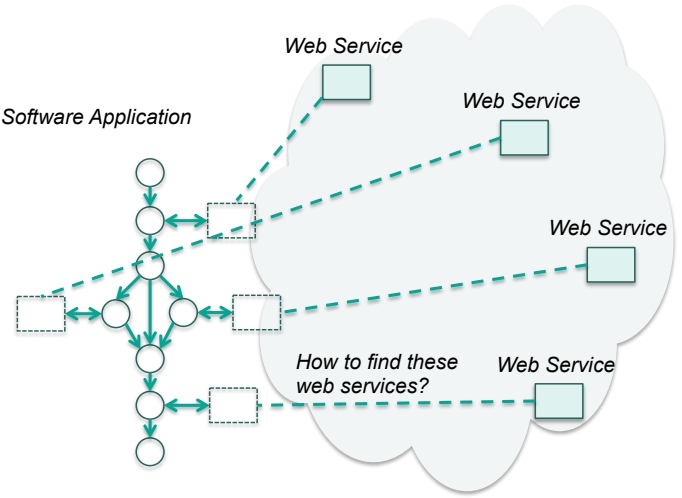


Figure 2.10: The use of web services for application development (composition)

(b) The web service directory is machine-readable and the application program itself is capable of conducting the service discovery process.

It should be clear that alternative (b) is far more sophisticated than alternative (a). It requires a formalized description of the service and a formalized search algorithm. It requires semantic in addition to syntactical methods to consider meaning and context.

Discovery Protocol

Figure 2.11 shows how the different actors in a web service environment work together [19]. The service provider has created a web service together with its description and publishes it in the web service directory. A service requester (application programmer) searches for a web service via the discovery interface and checks if the description matches his requirements. If an appropriate service is found, the programmer can invoke the service. Finally, the web service answers with a response message.

The fully automated web service engagement, as illustrated in Figure 2.11.b, is still a futuristic scenario. Even if the automated querying of service directory is possible, the understanding and interpretation of the search results and the selection of the appropriate web service matching based on functional and non-functional requirements (price, availability, etc.) still needs human intervention and judgment [20].

The ultimate vision is that web services discover other web services and interact with each other in a fully automated way and without any human intervention.

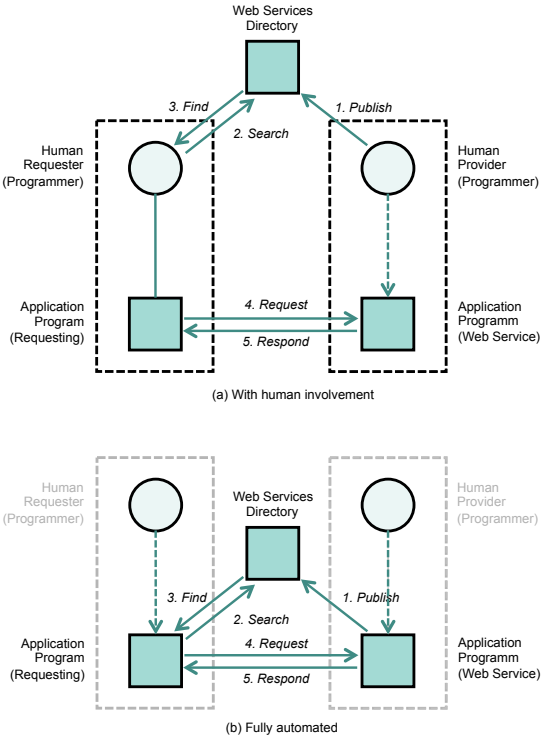


Figure 2.11: The process of engaging a web service

This vision has been described already in a 2001 Scientific American article by Berners-Lee, Hendler, and Lassila titled “The Semantic Web” [21]. Since then, the semantic web services research community has been working steadily to make this vision come true.

2.3.6 Previous Technologies

Previous technologies that covered the same objectives as web services, i.e., to enable two distributed applications to exchange data and call software functions included SUN Remote Procedure Call (RPC), the Common Object Request Broker Architecture (CORBA), Microsoft Distributed Component Object Model (DCOM), and Java Remote Method Invocation (JRMI). Table 2.2 shows the chronological introduction of each of these technologies over the years.

Listing 5.1 shows an example of an RPC protocol specification file that describes the remote version of the `PRINTACCOUNT` procedure. This specification can be seen as the precursor of WSDL. It contains the information needed for a client to invoke

Technology	Year
Sun RPC	1985
CORBA	1992
Microsoft DCOM	1996
Java RMI	1996

Table 2.2: Previous technologies to web services

a service or remote procedure.

```
/*
 * msg.x: Remote printing account protocol
 */
program ACCOUNTPROG {
        version ACCOUNTVERS {
                int PRINTACCOUNT(string) = 1;
        } = 1;
} = 0x20000099;
```

Listing 2.1: Example of the RPC interface definition language

R The differences and similarities between RPC and WSDL can be evaluated by contrasting the examples of interface definition languages given in Listing 5.1 and Figure 2.13

These technologies had drawbacks that were considered significant when developing distributed systems, especially in heterogeneous environments. For example, the RPC mechanism had mainly implementations for the UNIX platform and had, therefore, a reduce market share. CORBA and DCOM were fierce competitors which lead to a low acceptance and adoption from the industry. Furthermore, their programming was cumbersome. Finally, Java RMI technology was limited to the Java programming language.

These limitations were a clear indicator that a more open and consensual approach to develop distributed applications was necessary. Web services were the answer to allow software applications to easily communicate, independently of the underlying computing platform and language. The development of web services is substantially less complex than the prior solutions available for creating interoperable distributed systems.

R As a side note, CORBA already used the term service in its specification: “a service is basically a set of CORBA objects with IDL interfaces. The main characteristics of the objects composing a service is that they are not related to any application but are rather basic building blocks, usually provided by CORBA environments (e.g. transactions, naming, events).”

2.4 Web Service Technologies

The goal of web services is to ease the development of distributed systems. Services require to be autonomous and platform-independent, and need to be described, published, discovered, and orchestrated using standard protocols for the purpose of building distributed systems. The emphasis is on the definition of interfaces from a technical and programming perspective. The objective is automation and computerization, since web services provide a technological solution to enable transaction systems, resource planning systems, customer management systems, etc. to be integrated and accessed programmatically through the web.

(a) A web browser accessing a web service

(b) A software application accessing a web service

Figure 2.12: Remotely accessing web services

Nowadays, two types of web services can be identified:

- Operations-based web services.
- Resource-based web services.

2.4.1 Operations-based Web Services

The emergence of the internet forced components to expose their functionality to internal as well as external applications in the form of services. Since the preferred communication medium was the WWW, existing web standards and protocols were adopted and new ones were created, such as HTML, HTTP, XML, WSDL, SOAP, and UDDI, to support web services.

Definition — XML and HTML. XML (eXtensible Markup Language) is a specification language used to create schemas to share data on the web using standard

ASCII text. HTML (HyperText Mark-up Language) is a language to describe the presentation of a document so that it can be rendered in a web browser in a human readable form.

Definition — WSDL, SOAP, and UDDI. Three specifications used to develop distributed systems. The interface of services is described with WSDL, the web services description language; SOAP, originally defined as simple object access protocol, enables providers and clients to exchange messages and call software functions; and UDDI, an acronym for universal description, discovery and integration, is a directory where services are listed using WSDL.

A web service is an autonomous software component that is uniquely identified by a universal resource identifier (URI) which uses the hypertext transfer protocol (i.e., HTTP) to request and transport data as documents, the simple object access protocol SOAP to invoke remote functions, and languages, such as the extensible markup language XML, to structure data.

Figure 2.13 illustrates the skeleton of a WSDL interface definition that a client can use to invoke a remote operation provided by a web service. The service is called `Customer_Service` (line 1) and provides the operation `getAddress` (line 16-19) which accepts as input the message `CustIDRequest` (a `long`) and returns the message `AddressResponse` (a `string`) as output (lines 8-13 and 17-18).

SOAP-based web services achieve a loose coupling of distributed systems since a contract is specified between services using WSDL. It describes the operations (methods) a web service makes available to clients; which parameters operations receive and return; and which ports are used for communication. WSDL acts as a specification, very much like the interface definition languages used by RPC and CORBA middleware, to describe the interfaces, operations, and parameters of a web service. Separating the logical and technical layers leaves open the possibility to adopt different programming languages for the implementation of web services.

2.4.2 Resource-based Web Services

REST web services are resource-based services. The term REST refers to an architecture style for designing distributed applications, which uses the set of well-known HTTP operations `GET`, `PUT`, `POST`, and `DELETE` to change the state of remote resources. The underlying idea is that, rather than using complex mechanisms such as CORBA, RPC, or SOAP to connect applications, the simple HTTP protocol (and its associated methods) is used to interact directly with exposed resources. Unlike SOAP web services, it is not necessary to use fairly complex specifications such as SOAP itself. The main focus is on interacting with stateful resources, rather than operations (as it is with SOAP and WSDL). The communication is stateless. That is why REST services are often referred to as “stateless”.

```
1.  <definitions name="Customer_Service"
2.    targetNamespace="http://www.store.com/wsdl/CustService.wsdl"
3.    xmlns="http://schemas.xmlsoap.org/wsdl/"
4.    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
5.    xmlns:tns="http://www.store.com/wsdl/CustService.wsdl"
6.    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
7.
8.    <message name="CustIDRequest">
9.      <part name="CustID" type="xsd:long"/>
10.   </message>
11.   <message name="AddressResponse">
12.     <part name="address" type="xsd:string"/>
13.   </message>
14.
15.   <portType name="Cust_PortType">
16.     <operation name="getAddress">
17.       <input message="tns:CustIDRequest"/>
18.       <output message="tns:AddressResponse"/>
19.     </operation>
20.   </portType>
21.
22.   <binding name="Cust_Binding" type="tns:Cust_PortType">
23.   <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
24.   <operation name="getAddress">
25.     <soap:operation soapAction="getAddress"/>
26.     <input>
27.       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
28.         namespace="urn:store:getaddress" use="encoded"/>
29.     </input>
30.     <output>
31.       <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
32.         namespace="urn:store:getaddress" use="encoded"/>
33.     </output>
34.   </operation>
35.   </binding>
36.
37.   <service name="Cust_Service">
38.     <documentation>WSDL File for CustService</documentation>
39.     <port binding="tns:Cust_Binding" name="Cust_Port">
40.       <soap:address location="http://www.store.com/getAddress/">
41.     </port>
42.   </service>
43. </definitions>
```

Figure 2.13: Example of a WSDL interface definition

Definition — REST Service. An application-accessible web service that uses REST architectural principles and web specifications as underlying paradigms and technologies, respectively.

Roy Fielding [22] describes REST objectives in the following way: "The name Representational State Transfer (REST) is intended to evoke an image of how a well-designed web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."

Figure 2.14 shows an example of a REST request submitted as a URL to update an object (with several attributes) stored at Amazon AWS. The response to this

REST Request as a URL

```
1.  https://sdb.amazonaws.com/?Action=PutAttributes
2.  &DomainName=MyDomain
3.  &ItemName=Item123
4.  &Attribute.1.Name=Color&Attribute.1.Value=Blue
5.  &Attribute.2.Name=Size&Attribute.2.Value=Med
6.  &Attribute.3.Name=Price&Attribute.3.Value=0014.99
7.  &AWSAccessKeyId=your_access_key
8.  &Version=2009-04-15
9.  &Signature=valid_signature
10. &SignatureVersion=2
11. &SignatureMethod=HmacSHA256
12. &Timestamp=2010-01-25T15%3A01%3A28-07%3A00
```

REST Response

```
1.  <PutAttributesResponse>
2.    <ResponseMetadata>
3.      <StatusCode>Success</StatusCode>
4.      <RequestId>f6820318-9658-4a9d-89f8-b067c90904fc</RequestId>
5.      <BoxUsage>0.0000219907</BoxUsage>
6.    </ResponseMetadata>
7.  </PutAttributesResponse>
```

Figure 2.14: Example of a REST request and a response

action is an XML message with the return status. In contrast to operations-based web services, there is not formal definition of the remote interface. Nonetheless, and in this case, Amazon makes a web page available so that software developers know the parameters to use with REST requests.

2.5 Cloud Services

Web services provide a technological infrastructure that enables organizations to outsource computing resources as a service to support their business operations, including data storage, hardware, servers, and networking. These services are called *cloud services*. The term "cloud" is used to indicate that the service is remotely accessed using the internet. The cloud service provider owns the computing resources and is responsible for its acquisition, operation, and maintenance. The customer pays only on a per-use basis for the services used.

According to the US National Institute of Standards and Technology (NIST), cloud services are part of the cloud computing paradigm which is "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, and applications) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[4]

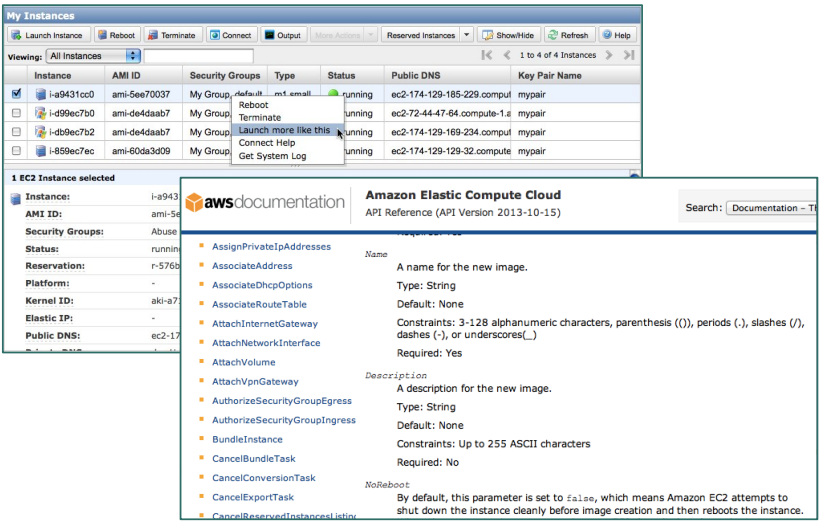
[4] `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`

Definition — Cloud Computing. The delivery of computing as a service rather than a product. Resources, software, data, and information are provided to customers, computers, and other devices as a utility (like the electricity grid) over a network (typically the internet).

Cloud computing enables consumers to establish a contract to use an application (a cloud service) hosted by the company that develops and sells the software. Common hosted solutions include enterprise resource planning (ERP) or customer relationship management (CRM) systems. The model does not require consumers to buy software licenses. The hosted cloud services give consumers more flexibility to switch providers and reduce the complexity and cost in maintaining the software.

Figure 2.15 shows the two types of services provided by Amazon Elastic Compute Cloud (EC2). On the left side (a), the screenshot in the background shows how EC2 can be accessed using the web console which is an electronic service accessible via the web. On the right side (b), the screenshot in the foreground shows the SOAP application-accessible web services made available to programmers.

(a) Electronic service interface of EC2



(b) Listing and description of web services made available by EC2

Figure 2.15: Services provided by Amazon AWS EC2

2.5.1 Economies of Scale

As in many areas of the industry, and society in general, new business models often distinguish themselves by bringing cost reductions when compared to existing solu-

tions. Cloud computing also fits nicely into this category. Compared to traditional enterprise data centers, there are large economies of scale that make this model a compelling alternative. Cloud services adopt the utility computing paradigm to provide customers on-demand access to resources in a very similar way to accessing public utilities such as water and energy.

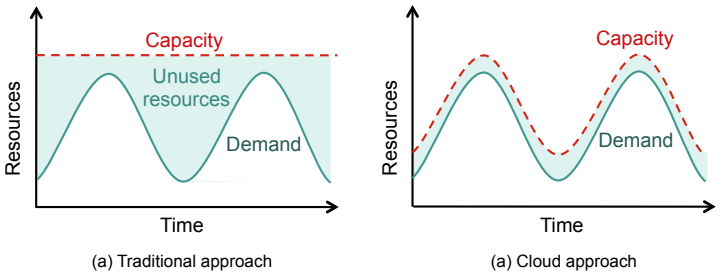


Figure 2.16: The economical model behind cloud computing

Figure 2.16 shows why cloud computing is an interesting economic solution. The left side of the figure illustrates one of the main drawbacks of the traditional approach that forced companies to invest in computing capacity to respond to occasional, high demand. This can typically be triggered by seasonal demand or to daily peaks of requests. This leads to a waste of capacity when demand is low since only a fraction of the computing infrastructure is utilized. The right side of the figure shows the cloud approach, where the company may request as much or as little computing resources as needed, and pays for these resources per use (pay-per-use model). On the provider's side, cost savings are achieved with a more efficient system administration, lower investments in infrastructure, and high levels of equipment sharing.

Administration In a large cloud computing center, one administrator can be responsible for several thousand servers which reduces operation costs. Additional cost savings are also made possible since many applications running in the cloud are accessed using a web browser. Browser-based applications require less administrative overhead on the customer side than traditional software since there is no need to install patches or upgrades. The provider is responsible for carrying out these tasks.

Infrastructure Owning infrastructure, equipment, and servers requires upfront capital costs. Cloud computing shifts high investments to service providers avoiding spending on hardware, software, and licensing fees. Customers can use software with minimal upfront costs using flexible pricing models such as pay-per-use.

Sharing Virtualization technologies allow many virtual servers to run on the same

physical machine. Servers, applications, and databases are uncoupled from physical hardware and presented as logical resources. This enables the rapid deployment of resources from a shared pool. Workloads share the infrastructure with other organizations' computing needs which leads to a reduction of costs.

Cloud computing is not without risks. Some issues have been pointed out. Standards are needed to ensure the interoperability of solutions so that cloud sevices, virtual images, applications, and tools can be moved across cloud environments without high engineering efforts. The transfer of data across cloud providers requires security, encryption, and privacy management. Cloud providers often advertise service levels but no solutions are in place for an effective monitoring and management of the quality of service rendered to customers (Chapter 10 will dive into service level engineering as an approach to make providers accountable when an insufficient quality of services is provided.)

2.5.2 Characteristics

Cloud computing is not a technology[5] but rather refers to a new business model based on a computer platform delivering on-demand services. Therefore, the characteristics of cloud services are not technical but refer to the particular set of features that is offered to consumers. NIST describes five typical characteristics that cloud services should exhibit.

On-demand self service enables customers to decide when to use resources and pay only when using them. A consumer can unilaterally request or release computing resources, such as processing time and data storage services, without requiring human interaction with service providers.

Broad network access allows services to be offered over the internet or private networks. Services are available from any computer, laptop, or mobile device.

Resource pooling enables customers to draw from a pool of computing resources usually located in remote data centers. Customers do not have to know where the resources are maintained. Several consumers can share the same resources which decreases final costs.

Rapid elasticity makes possible to scale services up or down to offer an adjustable number of resources. New resources can be added almost immediately when they are needed.

Measured services enable customers to pay only for the services used. The service provider has to measure, collect, and offer specific information about the services used to bill customers accordingly.

[5] Cloud services are usually provided via SOAP or REST web services but any other technology to support and implement distributed systems can be used.

For example, Amazon EC2 is a cloud service that provides elastic compute capacity. It provides a simple web-based self-service interface to obtain and configure capacity with minimal effort (Figure 2.15.a). Users can easily control computing resources such as boot new server instances, scale capacity – both up and down – as their computing requirements change.

2.5.3 Delivery Models

Cloud services can be categorized into three different main types (Table 2.3). Each type describes to which level a customer has control over the software, platform, and/or infrastructure.

The first level, *Software as a Service* (SaaS), enables consumers to use the provider's applications running on a cloud infrastructure. A popular example of SaaS is Google Docs (`docs.google.com`), an online productivity suite to create, manage, and share documents, spreadsheets, presentations, and surveys. The service provided by Google is accessible directly from a web browser and does not require software installation. Other well-known applications that also belong to this category include `dropbox.com`[6], `salesforce.com`, and `freshbooks.com`.

Delivery model	Examples
SaaS	`dropbox.com`, `docs.google.com`, `salesforce.com`, `freshbooks.com`.
PaaS	`heroku.com`, `appengine.google.com`, `force.com`.
IaaS	`rackspace.com`, `aws.amazon.com/ec2`, `windowsazure.com`.

Table 2.3: Cloud delivery models

The second level, called *Platform as a Service*, or PaaS, provides sophisticated platforms which can be used by customers to develop and run software applications. The provider often makes available programming languages, libraries, and tools so that customers can easily and quickly develop and compile new applications. Examples of PaaS include Google App Engine (`appengine.google.com`) which enables programmers to build and host web applications on Google's infrastructure; Heroku (`heroku.com`), a cloud platform to build and host web applications which support several programming languages (it was acquired by Salesforce in 2010); and `force.com`, the PaaS from Salesforce which enables customers to build new custom business apps to run on Salesforce's servers.

[6]Dropbox has a software module which requires its installation in a computer to use the service to synchronize files transparently

The lowest level, *Infrastructure as a Service* or IaaS provides physical servers, virtualised infrastructures, storage, networks, and other fundamental computing resources such as virtual-machine disk image library, block and file-based storage, firewalls, load balancers, IP addresses, and virtual local area networks. Servers can be fitted with any platform, operating system, and software applications.

Examples include Amazon EC2 (`aws.amazon.com/ec2`) which provides computing power, Windows Azure (`windowsazure.com`), Microsoft's cloud platform, and RackSpace (`rackspace.com`), a company which has built its solution on OpenStack, an open source cloud which does no lock customers into a specific private technology.

2.6 The Internet of Services

The term *Internet of Services* (IoS) appeared in 2007, introduced by SAP and supported by several research projects financed by the European Union. Nowadays, the term web of services is also being used with the same meaning. The IoS addresses the challenge of transforming services into "tradable goods" that are offered, sold, executed, and consumed via the web. The term service is used to identify services (e.g., human services, e-services, web services, and cloud services) provided through the web which are potentially linked and interconnected, in the same way that the web of pages or documents is connected.

Supporting the IoS requires models, platforms, and tools to make services tradable on the internet and composable into value-added services. Consumers select services from different providers based on their functionalities, best pricing, offered quality of service or rating. After selecting a service, it is delivered by its provider. Finally, the consumer will pay for the service consumption. This procedure is very similar to the operation of cloud services. The difference lies on the type of services provisioned.

Many research challenges around the mapping between services into the IoS are still unresolved. Some of the most critical and urgent questions which still need to be addressed by research and by developing new prototypes include service descriptions, service architectures, service level agreements, monitoring mechanisms, and computer processable legal terms, just to name a few.

2.6.1 Service Descriptions

Service marketplaces need to offer search mechanisms that allow for comprehensive search criteria. At the base for such mechanisms, a framework for describing different aspects of services is needed. A suitable *service description* framework covers not only the functional and technical description of a service but also aspects such as pricing, quality of service, user rating, and legal terms among others. Consumers need to be able to search for service functionality based on functional classifications such as UNSPSC, eClass, eOTD, Rosettanet Technical Dictionary

(RNTD), or natural language descriptions. The search results may then be further refined taking into consideration a large variety of non-functional properties.

A good example of achievements in the field of service descriptions is the Unified Service Description Language (USDL). It was developed in 2008 for describing business, software, or real world services using computer-understandable specifications to make them tradable on the internet [23]. Later, in 2011, based on experiment results from the first developments, a W3C Incubator group[7] was created and USDL was extended. In 2012, a new version named Linked USDL[8] based on semantic web technologies was proposed [24].

R Service descriptions are explored in more detail in Chapter 5 which explains how to enrich the description of cloud services with semantic knowledge. The enrichment is applied to a Web API built using the REST architecture style. The chapter also explains how semantics can contribute to develop more effective search algorithms.

2.6.2 Service Engineering

Methodologies, methods, reference models, and tools are required to enable a faster development of higher-quality, lower-cost services. *Service engineering* is an approach to the analysis, design, implementation, and testing of service-based ecosystems in which organizations and IT provide value for others in the form of services. Figure 2.17 shows a software workbench called ISE to engineer services. The prototype relied on various models, such as process models, organizational role, business rule, and data models to specify the behavior and technology required to design and implement a service.

Service engineering does not only provide methodologies to handle the increasing complexity of numerous business actors and their value exchanges, but it also provides reference models and tools for constructing and deploying services that merge information technology and business perspectives. Challenges include modeling, validation, and verification of services and their associated business processes, the smooth evolution and execution of business processes and the reliable management of services compositions.

2.6.3 Service Level Agreements

Prior to interacting with a service, the consumer can create a *service level agreement* (SLA) with the service provider stating the terms under which the service needs to be provided. Rights as well as obligations of both parties regarding the service consumption can be described. The aspects specified in a service level agreement (e.g., quality of service and pricing) need to be linked and derived from the service description as it provides the base for negotiation.

[7] `http://www.w3.org/2005/Incubator/usdl/`
[8] `http://linked-usdl.org/`

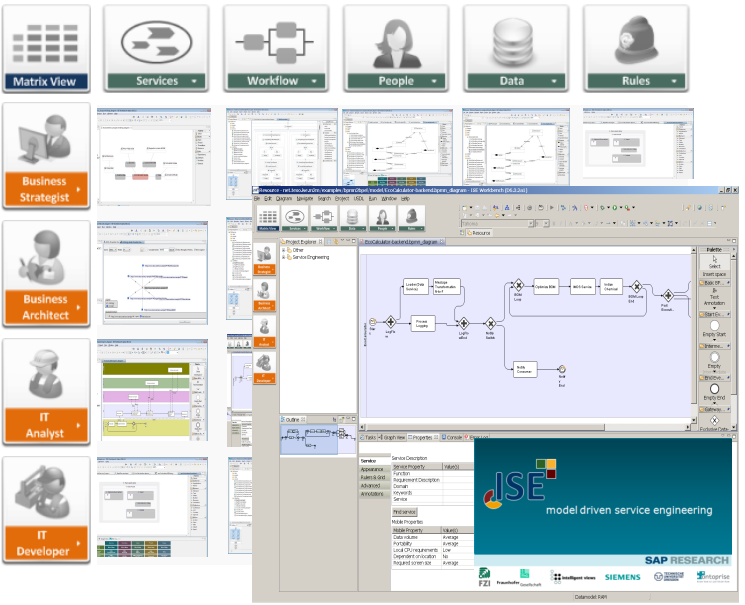


Figure 2.17: The ISE workbench to engineer services [25]

To enable trust among the participants, there is a requirement for monitoring SLAs and interactions. Creating monitoring environments for services requires mechanisms to display and analyze information flows between services participating in complex compositions to detect security risks and assess performance. Monitoring also needs to provide mechanisms to ensure trust and confidence in services created by end-users themselves. The goal is to make sure that service providers deliver services under the terms promised to the consumer. The monitoring of functionality may be provided by marketplaces or by trusted third parties. The base for the monitoring is the service level agreement negotiated between the provider and the consumer.

2.6.4 Business and Legal Models

To extract value from services, providers need appropriate business models since they enable to convert new technology into economic value. A special emphasis has to be given to the generation of new business models for all stakeholders (e.g., service providers, aggregators, and consumers) and corresponding incentive mechanisms. It is also an important determinant of the profits to be made from a service innovation and, in some cases, the innovation rests not in the service but in the business model itself [26].

The combination and integration of world-wide regulations and policies is

fundamental when provisioning services to end consumers. Legal aspects are subject to extensive government regulations. In European countries, regulation is a combination of central and local controls. Frameworks are needed to facilitate the reasoning about IoS ecosystems across their geographic, economic, social, and legal dimensions. Crafting an appropriate and customized legal framework will help building a service economy that is as robust as existing economies for manufactured goods, commodities, and human-provided services. Technical and legal mechanisms which promote law-abiding attitudes need to be studied.

2.7 Conclusions

This chapter presented two distinct perspectives used to characterize the evolution of services over the past 50 years: (1) the automation of economic activities and self service and (2) the improvement of a programming paradigm.

The first perspective focuses on the creation of electronic services and identifies services from an economic perspective. The aim of this perspective was to reduce the cost of providing services by replacing humans by automated machines. For example, undertaking a trip by train has traditionally required passengers to purchase a ticket from an office and show it for inspection when required by the train operator. As a response to technological development, automatic dispensers and on-line services accessed with web browsers have reduced the cost of service provisioning.

The second perspective looks into services from a computer science view and led to the development of web services and cloud services. These services resulted from the adoption of standards and unified interfaces to enable the interoperability of heterogeneous components to truly support distributed systems. Services, such as web services, use specifications, protocols, and interfaces to enable remote software applications to communicate. Computers located anywhere in the world can request for services to store data, send e-mails, perform complex computations, or encrypt documents.

Review Section

Review questions

1. ATM machines were one of the first electronic services to be developed in the late 60s. Identify other electronic services introduced in the 70s and 80s.
2. Use Froehle and Roth [3] classification to characterize the following services: `expedia.com`, `booking.com`, `99designs.com`, `redbeacon.com`, `lulu.com`, `threadless.com`, `odesk.com`, and `facebook.com`. Give additional examples to cover all types of the classification.
3. Identify existing technology-free, -assisted, and -facilitated services which

can constitute good candidates for their transformation into technology-mediated and -generated services.

4. Find programming examples of applications implementing the client-server model using RPC, CORBA, DCOM, and JRMI. The client–server model is a distributed application structure that partitions tasks between the service providers (requested service) and service requesters (requesting application). Contrast the benefits and difficulties of each programming techniques.
5. The interface of a SOAP web service is described with the specification language WSDL. Provide an example of a WSDL description of a web service with two operations: `string getAddress(long custID)` and `setAddress(long custID, string regionID)`.
6. Which benefits can cloud services provide to businesses? What type of cost savings can be achieved? How flexible and agile are cloud services? Who typically owns the data and where it is stored?
7. Classify the following cloud services as SaaS, PaaS, or IaaS: `lunacloud.com`, `scalextreme.com`, `cirrhus9.com`, `logicworks.net`, `cohesiveft.com`, and `appcore.com`.
8. Contrast and compare web services, cloud services, and Internet of Services.

Project

This project analyzes the cost differences of deploying an on-premise physical and software infrastructure versus adopting services from a cloud computing provider. You will use the calculator provided at `tco.2ndwatch.com` to compare the total cost of ownership (TCO) of both approaches and highlight key points when considering cost. The total cost of ownership accounts for the costs to run a software system over its lifetime. It is the best metric to compare the costs of cloud computing and on-premise software deployments. It includes the fees paid to vendors, maintenance and support, and hardware, equipment, and staff costs. The service `tco.2ndwatch.com` calculates the TCO of using Amazon Web Services versus running applications on on-premise infrastructures.

In a first step, select a company or organization you are familiar with (e.g., a university, library, or research center) and make a comprehensive description of its ICT needs, staff, operations, and infrastructure. Afterwards, estimate the following parameters which are used by `tco.2ndwatch.com` to calculate the TCO (the web site of the service provides additional information on each parameter):

- Web application servers
- Database servers
- Overall storage
- Data centers
- Growth rate

- Administrative overhead
- Usage pattern

Download the report generated and examine the total expenditures for both strategic approaches. Which one is more cost-effective? What are the main reasons? What characterizes the borderline which can make one of the approaches more attractive over the other?

In a second step, use the service provided at `planforcloud.com` to determine which cloud provider would supply the most cost-effective solution for the company under study. What are the reasons?

In a last step, write a concise expert report with all the findings recommending to a (possible) manager the best approach to follow (cloud computing or on-premise) and, if a cloud computing approach is recommended, which cloud provider would be best suited to contract.

Key terms

Electronic Service An electronic service, or shortly e-service, is a service that allows a remote interaction using information and communication technologies (ICT) such as the internet, software applications, and computing resources.

Web Service A web service is a technology and approach of communication which enables a software system to support interoperable machine-to-machine interaction over a network.

SOAP Service A SOAP service is an application-accessible web service that uses the SOAP protocol for exchanging structured information between the two parties involved, i.e., the service provider and the service client.

REST Service A REST service is an application-accessible web service that uses REST architectural principles and web specifications as underlying paradigms and technologies, respectively.

Cloud Service Cloud services are designed to provide easy, scalable access to applications and resources. They are managed by cloud service providers. Services are made available on-demand from a cloud computing provider's servers in contrast to being provided from a company's own on-premise servers. Popular cloud services include Google Docs (documents), Dropbox (files), and Flickr (photos).

Cloud Computing Cloud computing refers to the delivery of hosted services over the internet (i.e., the cloud). Services are predominantly divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

Service-Oriented Architecture An architectural style and business-centric programming paradigm to develop distributed systems where systems consist of software clients, which act as service consumers, and software providers,

which act as service providers.

Internet of Services The internet of services envisions to provide an ecosystem to foster the trading of application and human services over the internet. Beyond downloading music, ordering books, storing files remotely, and booking flights, services can also be traded as commodities.

Service Descriptions Service descriptions are generally formal representations of functional and non-functional characteristics of services. SOAP web services use WSDL, and electronic services can use Linked USDL for their descriptions.

Further reading

Olaf Zimmermann, Mark Tomlinson, and Stefan Peuser. *Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects*. Springer, 2013.

Leonard Richardson, Mike Amundsen, and Sam Ruby. *RESTful Web APIs*. O'Reilly Media, 2013.

Thomas Erl, Ricardo Puttini, and Zaigham Mahmood. *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall, 2013.

Jorge Cardoso and Amit Sheth. *Semantic Web Services, Processes and Applications*. Springer, 2006.

References

[1] Victor Fuchs. *The Service Economy. National Bureau of Economic Research*. 1968 (cited on page 38).

[2] Daniel Castro, Robert Atkinson, and Stephen Ezell. *Embracing the Self-Service Economy*. 2010 (cited on page 39).

[3] Craig Froehle and Aleda Roth. "New measurement scales for evaluating perceptions of the technology-mediated customer service experience". In: *Journal of Operations Management* 22.1 (2004), pages 1–21. ISSN: 0272-6963 (cited on pages 40, 44, 72).

[4] Roland Rust and Prem Kannan. "E-service: a new paradigm for business in the electronic environment". In: *Communications of the ACM* 46.6 (2003), pages 36–42 (cited on pages 44, 47).

[5] Jennifer Rowley. "An analysis of the e-service literature: towards a research agenda". In: *Internet Research* 16.3 (2006), pages 339–359 (cited on page 44).

[6] *EU directive 2006/123/EC of the European parliament and of the council of 12 December 2006 on services in the internal market.* Technical report. European Union, 2004 (cited on page 44).

[7] Charles Hofacker et al. “E-Services: A Synthesis and Research Agenda”. In: *Journal of Value Chain Management* 11.2 (2007), pages 13–44 (cited on pages 44, 47).

[8] *The e-government imperative: main findings*. Technical report. OECD, 2003, pages 1–8 (cited on page 45).

[9] Ziv Baida et al. “A Shared Service Terminology for Online Service Provisioning”. In: *Proceedings of the Sixth International Conference on Electronic Commerce (ICEC04)*. ACM Press, 2004 (cited on page 45).

[10] Jari Vesanen. “What is personalization? A conceptual framework”. In: *European Journal of Marketing* 41.5/6 (2007), pages 409–418 (cited on page 47).

[11] Ruth Bolton and Shruti Saxena-iyer. “Interactive Services: A Framework, Synthesis and Research Directions”. In: *Journal of Interactive Marketing* 23.1 (2009). Anniversary Issue, pages 91–104 (cited on page 47).

[12] Iris Junglas and Richard Watson. “Location-based Services”. In: *Commun. ACM* 51.3 (Mar. 2008), pages 65–69. ISSN: 0001-0782 (cited on page 47).

[13] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion Books. Hyperion, 2008. ISBN: 9781401309664 (cited on page 48).

[14] *Public Services Online: Digital by Default or by Detour?* Technical report. European Commission, 2013. URL: `http://ec.europa.eu/digital-agenda/` (cited on page 51).

[15] D. J. Wheeler. “The use of sub-routines in programmes”. In: *Proceedings of the 1952 ACM national meeting*. 1952, page 235 (cited on page 52).

[16] D. L. Parnas. “On the Criteria to Be Used in Decomposing Systems into Modules”. In: *Commun. ACM* 15.12 (Dec. 1972), pages 1053–1058. ISSN: 0001-0782. DOI: `10.1145/361598.361623`. URL: `http://doi.acm.org/10.1145/361598.361623` (cited on page 53).

[17] Mike Papazoglou and Dimitrios Georgakopoulos. “Introduction: Service-oriented Computing”. In: *Commun. ACM* 46.10 (Oct. 2003), pages 24–28 (cited on page 54).

[18] Herbert Simon. “The architecture of complexity”. In: *Proceedings of the American Philosophical Society*. 1962, pages 467–482 (cited on page 55).

[19] David Booth, Francis McCabe, and Michael Champion. *Web Services Architecture - W3C Working Group Note*. Technical report. Feb. 2004 (cited on page 58).

[20] Ethan Cerami. *Web Services Essentials*. Edited by Simon St.Laurent. 1st. Sebastopol, CA, USA: O’Reilly & Associates, Inc., 2002. ISBN: 0596002246 (cited on page 58).

[21] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (May 2001), pages 34–43 (cited on page 59).

[22] Roy Thomas Fielding. “Architectural Styles and the Design of Network-based Software Architectures”. PhD thesis. Irvine, California: University of California, Irvine, 2000 (cited on page 63).

[23] Jorge Cardoso et al. “Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments”. In: *IEEE International Conference on Services Computing (SCC)*. Florida, USA, 2010, pages 602–609 (cited on page 70).

[24] Carlos Pedrinaci, Jorge Cardoso, and Torsten Leidig. “Linked USDL: A Vocabulary for Web-Scale Service Trading”. In: volume 8465. LNCS. Springer, 2014, pages 68–82 (cited on page 70).

[25] Jorge Cardoso et al. “IoS-Based Services, Platform Services, SLA and Models for the Internet of Services”. In: *Software and Data Technologies*. Volume 50. Communications in Computer and Information Science. Springer, 2011, pages 3–17 (cited on page 71).

[26] Henry Chesbrough and Richard Rosenbloom. “The Role of the Business Model in Capturing Value from Innovation: Evidence from Xerox Corporation’s Technology Spin-Off Companies”. In: *Social Science Research Network Working Paper Series* (May 2002) (cited on page 71).