8

Service-Oriented Business Intelligence

Alberto Abelló and Oscar Romero

Department of Service and Information System Engineering
Universitat Politècnica de Catalunya, BarcelonaTech
Barcelona, Spain
{aabello,oromero}@essi.upc.edu

Summary. The traditional way to manage Information Technologies (IT) in the companies is having a data center, and licensing monolithic applications based on the number of CPUs, allowed connections, etc. This also holds for Business Intelligence environments. Nevertheless, technologies have evolved and today other approaches are possible. Specifically, the service paradigm allows to outsource hardware as well as software in a pay-as-you-go model. In this work, we will introduce the concepts related to this paradigm and analyze how they affect Business Intelligence (BI). We will analyze the specificity of services and present specific techniques to engineering service systems (e.g., Cloud Computing, Service-Oriented Architectures -SOA- and Business Process Modeling -BPM-). Then, we will also analyze to which extent it is possible to consider Business Intelligence just a service and use these same techniques on it. Finally, we store the other way round. Since service companies represent around 70% of the Gross Domestic Product (GDP) in the world, special attention must be paid to their characteristics and how to adapt BI techniques to enhance services.

Keywords: Services, Business Intelligence, Service-Oriented Architectures, Business Process Modeling.

8.1 Introduction

As defined in [26], "*services are economic activities offered by one party to another, most commonly employing time-based performances to bring about desired results in recipients themselves or in objects or other assets for which purchasers have responsibility. In exchange for their money, time and effort, service customers expect to obtain value from access to goods, labor, professional skills, facilities, networks, and systems; but they do not normally take ownership of any of the physical elements involved.*" In [40], we find a much simpler way to identify what a service is: The "Unified Services Theory" states that all managerial themes unique to services are founded in customers providing significant inputs into the production process. In [44], it is explained that emerging services emphasize economies of knowledge and adaptiveness,

M.-A. Aufaure and E. Zimányi (Eds.): eBISS 2011, LNBIP 96, pp. 156–185, 2012.
© Springer-Verlag Berlin Heidelberg 2012

shifting the focus from mass production to mass customization. This shifting is intended to provide superior value by meeting their unique needs for services. Thus, in this work we will analyze the specificity of this sector and present specific techniques to manage and engineer their systems (e.g., Cloud Computing, Service Oriented Architectures and Business Process Modeling). Then, we will analyze to which extent Business Intelligence (BI) can be regarded or even managed as a service and use these same techniques in its engineering methods.

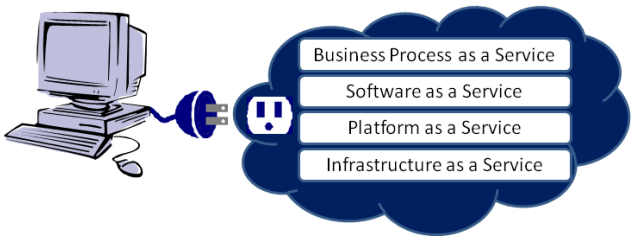

Fig. 8.1. Kinds of services

Services can be provided, mainly, at four different levels. As depicted in Figure 8.1, these are different layers built one on top of the other. Firstly, we can virtualize hardware and provide memory, CPU, disks, etc. This is known as Infrastructure as a Service (IaaS, Section 8.2) and, for example, Amazon's EC2 is a first-class IaaS citizen. Specific benefits of IaaS are fast provisioning and scaling, while the customer still controls the software. As a disadvantage, development and maintenance of applications remains at the customer side. Above this, we find Platform as a Service (PaaS, Section 8.3), where basic software (like Database Management Systems -DBMS- or programming platform) is provided (such as, for example, Google Application Engine). The benefit obtained at this level is that the environment is already installed and periodically maintained, while customer still keeps control of the application. On the other hand, some constrains may appear on what can be installed and porting the application to another service provider may not be easy. At the third level, we can also consume Software as a Service (SaaS, Section 8.4). Now, not only the platform, but also the whole application is installed and maintained by the provider (e.g., Google Apps). On the contrary, now we are tightly tied to the provider and it will be even harder to move to another provider. At the top, we could contract the implementation of parts of our business processes, i.e., Business Process as a Service (BaaS, Section 8.5). In this case, we are externalizing part of the business (not just the software). Well known examples are PayPal, iPhone App store, Chrome web store, etc. Some advantages come now in the form of flexibility, scalability and scale economy, but the relationship with the service provider is of complete dependency.

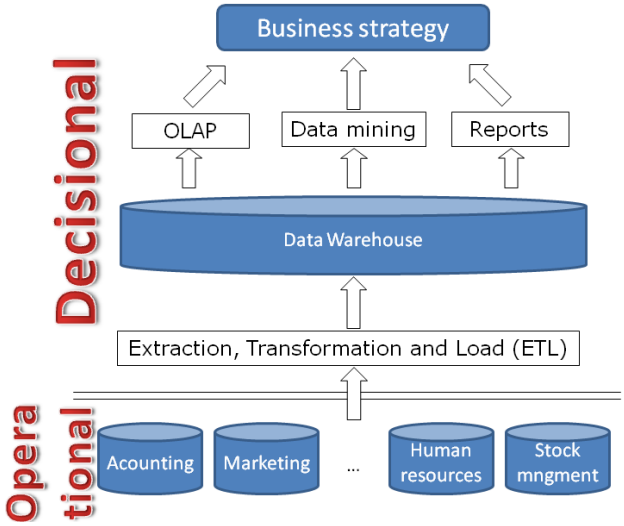

Fig. 8.2. Business Intelligence data flow

Business Intelligence (BI) is the process of making informed decisions in the company. As depicted in Figure 8.2, the central component of a traditional BI system is the Data Warehouse (DW) that stores all relevant historical data in an integrated way. Extraction, Transformation and Load (ETL) tools are in charge of feeding the DW mainly from operational systems or external sources (e.g., through the Internet). On the other side, we find analytical tools that allow users to retrieve data in different ways. The main families of tools are On-Line Analytical Processin (OLAP), Data Mining, and Query and Reporting.

BI systems can benefit from services at any of those four levels in Figure 8.2. In this work, we focus on how to exploit it. In principle, service engineering could not seem feasible for analytical, on-the-fly queries and tasks (e.g., given the amount of data handled by these systems, the highly customizable degree required and the difficulty to outsource processes such as ETL). Furthermore, from the provider point of view, offering BI as a service might be wrongly understood as lowering the barriers to enter the software business (which use to be lower than those in manufacturing, as pointed out in [40]). However, this is not the case. Economies of scale (by easily offering the same service to many consumers through the Internet), capital requirements (for the IT infrastructure needed underneath), and proprietary technology (that can easily evolve) yet fortify those barriers in this case. Indeed, service science happens to be an interesting shot for BI given some of its main characteristics. Interestingly, service oriented systems are described as highly

customizable and user-centered, which suits BI. However, the core difficulties regarding BI service oriented approaches lay on automating the process logic units that would conform the resulting BI system. In section 8.6, you will find a discussion of pros and cons of BI as a Service.

Furthermore, given that service companies represent around 70% of the Gross Domestic Product (GDP) in the world, it is definitely valuable to study and analyze those major drawbacks service oriented systems should face. Thus, we also propose to have a look the other way round and pay special attention to how BI characteristics and techniques can be applied to improve service oriented systems, which is discussed in Section 8.7.

8.2 Infrastructure as a Service (IaaS)

One problem in data analysis tasks is the vast amount of resources they consume. Nevertheless, we do not launch analytical processes day after day or, at least, not of the same size. Thus, a company should provision computer force for the peaks of critical load or for urgent tasks that some day may need. Besides the difficulty to predict when such tasks may arrive and how many resources they would demand, most of the time those provisioned resources will be wasted, because of lack of analytical tasks, or because they just do not need the maximum computer capacity. This waste is directly translated into a loss of money for the company.

A well known e-business, pointed out in [18], is shared IT infrastructure. As defined by NIST (National Institute of Standards and Technology) in [29], IaaS allows for the provisioning of fundamental computing resources (like processing, storage, network, etc.) to deploy and run arbitrary software, which can include operating systems and applications. These services allow a consumer to request and receive a new computer instance without needing to focus on IT concerns such as network placement and hardware availability.

A promising technology for reaching IaaS is *Cloud Computing*, which is defined in [29] as "*a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*". Nowadays, two kinds of clouds are distinguished: private and public ones. *Private clouds* are those data centers that belong to a company and are just used by its departments. On the contrary, *public clouds* are those providers of *utility computing* to the public in a pay-as-you-go manner. In this sense, *utility computing* must be understood as the service sold by public clouds. This perfectly suits the characteristics and needs related to analytical tasks. Instead of buying an expensive machine or data center, we can just *pay-as-we-go*. Thus, throughout this paper, we will use the term "cloud" to refer to public clouds providing utility computing.

Also in [29], we find the essential characteristics of Cloud Computing:

On-demand self-service: Customers being able to provision computing capabilities without requiring human interaction.
Broad network access: Utility computing can be consumed from heterogeneous client platforms (e.g., laptops, PDAs, etc.) anytime and anywhere.
Resource pooling: Computing resources are pooled to serve multiple consumers.
Rapid elasticity: Capabilities can be rapidly and elastically provisioned (they appear to be unlimited from the consumer's point of view).
Measured service: Resource usage can be monitored, controlled and reported (allowing a pay-per-use business model).

Some of these characteristics are emphasized in [48], which, from a hardware point of view, considers that there are three new aspects in Cloud Computing:

1. The illusion of infinite computing resources available on demand (i.e., characteristics "On-demand self-service" and "Rapid elasticity"). In other words, there is not maximum limit.
2. The elimination of an up-front commitment by cloud users (i.e., "Rapid elasticity"). In other words, there is no minimum commitment.
3. The ability to pay per use of computing resources on a short-term basis (i.e., "Measured service").

Specially "Rapid elasticity" and "Measured service" characteristics look like specially interesting for BI, where the load of analytical tasks is not constant or even predictable at mid term. In [40], the authors outline the importance of JIT (Just In Time) information, for which "Broad network access" is a desirable property. Furthermore, it can be better achieved by the elastic power of parallel computing in the cloud (for example, see [50]). Since you are charged by processor cycles being used, you can parallelize tasks as much as possible (you have *"infinite"* available machines), to finish your processing as soon as possible. Provisioning physical machines for the maximum level of parallelism you may ever want to enjoy is clearly unaffordable.

From the economic point of view, Cloud Computing could offer services below the cost of data centers and still make profit thanks, for example, to the difference in the price of energy. A data center can be billed one million euros per year in electricity, whose cost highly varies from country to country depending on the natural resources each has. This cost is mainly due to refrigerating processors, which clearly depends on the latitude of the location. As pointed out in [5], it is cheaper to ship data over fiber optic that to ship electricity over high-voltage transmission lines.

On the other hand, [40] underlines the problem of delaying the delivery of the service, i.e., customers expect to be served shortly after they demand the service. This can be solved by increasing service capacity, which leads to private data centers being clearly underused (studies indicate that they

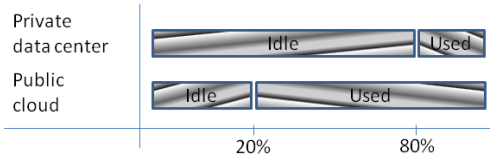

Fig. 8.3. Capacity usage in private data centers and public clouds

remain idle more than 80% of the time), while the usage of a cloud is estimated between 60% and 80% (see Figure 8.3 for a sketch and [5] for details), thanks to multiplexing many tasks. Well known generic strategies for capacity management, as explained in [18], can be used, namely *level capacity* (i.e, *customer-induced variability*, *segmenting demand*, *offering price incentives*, *promoting off-peak demand*, and *reservation and overbooking*) and *chase demand* (i.e., *creating adjustable capacity*, and *sharing capacity*).

In [1], the authors also find Cloud Computing specially appropriate for data intensive applications. Relevantly, it remarks that we only benefit from elasticity if (i) the workload is parallelizable, (ii) data are stored in an untrusted host, and (iii) data are replicated (often across large geographic distances). Going into more detail, ten obstacles (and also research opportunities) for IaaS are detected in [5]. Let us briefly analyze them from the point of view of BI:

Availability of Service: This issue is really important for transactional processing, since a service not available directly affects the number of clients of our company. Nevertheless, for analytical activities this may not be so critical.

Data Lock-In: A problem related to the usage of IaaS is that data must be moved in and outside the provider infrastructure, which usually forces the usage of a proprietary API. If our data sources are not already inside the cloud, we will probably need specific programs to move data in.

Data Confidentiality and Auditability: There should not be any problem to make a cloud as secure and reliable as possible (e.g., anonimization functions and encrypting). Nevertheless, some companies may feel reluctant to give their sensitive data, which, in turn, happens to be the most valuable source of knowledge in most cases. Moreover, some legal issues also have to be taken into account. For example, contracts can explicitly limit the movement of data within national boundaries.

Data Transfer Bottlenecks: Internet transfers are slow. Given that cloud providers charge per data movement (they count bytes transferred) and also the large amounts of data in analytical tasks, this may be a problem (generating extra costs that have to be considered).

Performance Unpredictability: Cloud providers schedule the tasks in their machines, as explained above, in order to level capacity (either memory, disk or any other resources). Thus, if we want our analytical job to be

prioritized it should be so stated in the contract, taking into account variations in the average performance of memory access are smaller than those in the average performance of disc access, for example.

Bugs in Large-Scale Distributed Systems: Debugging is not easy in cloud applications. This may be a problem if we are developing an ad-hoc analytical application, given its complexity and high execution cost.

Scaling Quickly: Given that analytical tasks are data as well as computation intensive, they can present special problems to the provider to absorb their load without violating the service contract.

Scalable Storage: The illusion of infinite resources is easier to be obtained for processors than for storage. In data warehousing, we do not need to scale down, but only to scale up, which simplifies the management and provisioning problem, from the provider point of view.

Reputation Fate Sharing: Laws exist in each country to restrict data management. It should be clearly stated who is responsible (either cloud providers or consumers) for analytical tasks breaking these laws.

Software Licensing: Not only hardware, but also software costs have to be taken into account (charging an annual license does not make sense any more). The provider transfers the cost of the service to the consumer, but it has to be done based on the usage. This problem does not look like having special consequences in the case of BI, beyond those in other kinds of applications.

Finally, according to [2], several challenges appear on having database applications in the infrastructure of a cloud, which are relevant to us, since BI tasks are data intensive:

- Deployment
  - Localization (i.e., generate IP addresses for the virtual machine of each DBMS instance).
  - Routing (i.e., route application requests to the appropriate DBMS instance).
  - Authentication (i.e., be aware of the credentials of all clients independently of where it is run in the cloud).
- Tuning
  - Placement (i.e., mapping virtual machines to physical machines).
  - Resource partitioning (i.e., distribute the resources of the physical machine among the virtual machines running on it).
  - Service level objectives (i.e., be able to minimize resources usage, while maintaining adequate performance).
  - Dynamically varying workloads (i.e., deal with changes in the workload, potentially defining classes and moves from one class to another).

Summing up, the main benefit of IaaS for BI is the rapid elasticity of resources, which is relevant in this kind of applications given the amount of

data and unpredictability of queries. On the other hand, the main problem is that the customer has to trust the provider with his/her data.

8.3 Platform as a Service (PaaS)

As defined in [29], PaaS allows to deploy consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying infrastructure including network, servers, operating systems, or storage (like in IaaS) but platform service offerings include workflow facilities for design, development, testing, deployment, and hosting, as well as services that enable team collaboration, web service integration and marshalling, database integration, security, scalability, storage, persistence, state management, application versioning, application instrumentation, and developer community facilitation. The consumer, in turn, has control over the deployed applications and possibly application hosting environment configurations. These services are provisioned as an integrated solution over the Web.

Cloud Computing does not necessarily lay under PaaS, but it clearly fits. In general, any application or platform can run in a cloud. For example, the authors of [36] contend that we have to start thinking about data management as a service. In the case of DBMSs, it is well known (see [12] and [32]) that the shared-nothing architecture of a cloud easily fits. Nevertheless, some modifications must be introduced to benefit from scalability ([43] notes that differences come from implementation choices and not from fundamental differences in the model). Thus, let us divide the kind of platforms we may find to support BI in two main groups: those made available in a cloud using a special version (normally better exploiting parallelism) or license (i.e., *software to use in a cloud*, which were born outside the clouds and then ported) and those implying cloud and parallel computing underneath (i.e., *cloud software*, which were born to be used in a cloud).

8.3.1 Software in a Cloud

Cloud Computing offers new markets and software vendors are well aware of this. Some examples (not intended to be exhaustive) of tools offering a BI product or version to be executed in Massively Parallel Processing (MPP) systems (i.e., clouds) or just a platform to run or BI development applications are:

- Vertica: A column store, descendant of the open source project C-Store in [42].
- Infobright: An analytic column-oriented database designed to handle business-driven queries on large volumes of data (stored in a grid) - without IT intervention.
- nCluster: Hybrid row and column DBMS.

- K2Analytics: A Managed Service Provider (MSP), which houses the customers licensed software.
- LogiXML: A code-free application development environment including hundreds of pre-built elements, that reduce the amount of work needed to create and maintain more complex BI applications.

8.3.2 Cloud Software

Despite any application can run in a cloud, some projects appeared lately that propose new architectures and algorithms to exploit its specific characteristics and possibilities. Systems in this category use to be tagged with the "NOSQL" buzzword (standing for Not Only SQL). Nevertheless the difference is not whether they provide an SQL interface or not, but the data model they follow, which is not relational. Thus, the temptation might be to name them "NORelational". Nevertheless, as explained in [28], "CoRelational" would be more appropriate, as they could be better understood as complementary.

These systems are based on the storage and retrieval of key/value pairs (as opposed to the relational model based on foreign-key/primary-key relationships). Relational referential integrity assumes a closed world (where all data are known, as opposed to an open world), which is hard to be assumed in a cloud or the Internet. While in the Relational model child nodes point to the well known parent (by means of a foreign-key), in the CoRelational model are parents who point to their children (whose information may not exist beyond the parent limits). Thus, instead of having each sale pointing to the corresponding product, date, customer, etc., we keep a lists of pointers indexed by the product, date and customer key (i.e., we reverse the pointers). It is in this sense that [28] compares both models under the prism of *category theory* to conclude that they are actually dual.

In the following subsections we just summarize the main concepts of BigTable and MapReduce as they were presented by Google in [9] (open source implementation [4]) and [11] (open source implementation [3]), respectively (another interesting Google product, we are not going to analyze, is Dremel, for details of the latter see [30]).

BigTable

is a distributed storage system designed to scale to very large size (petabytes). Figure 8.4 sketches data organization inside it, whose main structure is [key,value] pairs. The main characteristics we would like to outline here are:

- Data are indexed by row and column values (which are arbitrary strings).
- Columns can be grouped into families to be physically stored together.
- Versions are automatically generated for each value (which are timestamped).
- Data are treated also as uninterpreted strings.

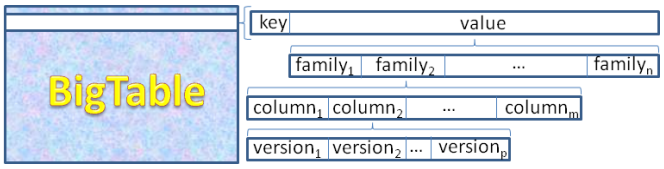

Fig. 8.4. BigTable organization

- Only single-row transactions are supported.
- Data are clustered (i.e., physically sorted) by key.

Note that only families of columns are part of the schema and have to be stated on creating a table. Oppositely, the columns are dynamically defined on inserting data. A family actually corresponds to a separate storage. Regarding data retrieval, it provides random access to one key, as well as parallel scan of the whole table or a range of keys.

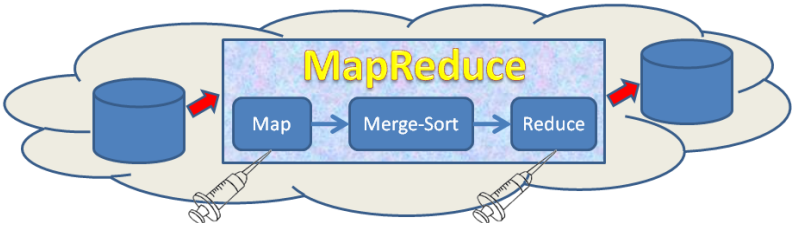

Fig. 8.5. MapReduce overview

MapReduce

Is a programming framework that allows to execute user code in a large cluster. It hides parallelization, data distribution, load balancing and fault tolerance from the user. All the user has to do is writing two functions: Map and Reduce. As sketched in Figure 8.5, those functions are injected in the framework.

Thus, the signature of those two functions is as follows:

$$map(key_{in}, val_{in}) \rightarrow \{[key^1_{tmp}, val^1_{tmp}], ..., [key^n_{tmp}, val^n_{tmp}]\}$$

$$reduce(key_{tmp}, \{val^1_{tmp}, ..., val^m_{tmp}\}) \rightarrow \{[key^1_{out}, val^1_{out}], ..., [key^p_{out}, val^p_{out}]\}$$

The execution would be:

1. Map function is automatically invoked for each pair [key,value] in the source table (it also works for plain files, in which case, one pair is generated per line, having as key the position inside the file). Notice that the source table is distributed in a cloud. Therefore, the framework takes advantage of this and tries to execute each map call locally to the data. Each call can generate either one new pair (potentially different from that in the input), many pairs or none at all.
2. The intermediate pairs [key,value] generated by the different executions of the Map function, which are temporally stored in the distributed file system, are then ordered using a distributed merge-sort algorithm.
3. For each different intermediate key, the Reduce function is invoked once receiving together all values associated to it. Each call can generate again either one new pair (also potentially different from that in its input), many pairs or none at all.

In [35], the authors compare and analyze the performance of Hadoop (which is an open source MapReduce implementation) against that of two parallel DBMSs. They conclude that DBMSs outperform Hadoop, while Hadoop is more tolerant to software/hardware failures. Reasons for these differences are carefully analyzed and justified to conclude, as in [43], that both technologies are rather complementary. [23] identifies five design factors that affect the performance of Hadoop and shows how by tuning these factors its performance improves to be almost comparable to that of the DBMSs. Another problem of working in a distributed environment (already pointed out in previous section) is that of debugging. KarmaSphere offers a development environment for Hadoop that allows to monitor the execution and debug your code.

Generalizations of MapReduce are presented in [46] and [6]. The latter extends the framework to allow other kinds of operations (i.e., Map and Reduce are just special cases of the operations that can be injected by users). Special attention is devoted to operations with more than one data input stream (e.g., join, which results to be really hard and artificial to be obtained in MapReduce).

Both technologies fit naturally to advanced BI tasks. For example, [21] argues that next data warehouse generation must consider all kind of data available, which includes textual data (not only plain text files but others such as mails) and unstructured sources (generally coming from the Web). The immediate consequence is that the amount of data to be stored grows exponentially and accordingly do so the size of the analytical tasks to be performed. BigTable and MapReduce are two promising paradigms to tackle, respectively, both consequences. Whereas BigTable scales to petabytes, the MapReduce paradigm transparently (i.e., the framework is responsible) provides grouping and ordering (essential for analytical tasks) regarding the key values produced.

All in all, the benefit of PaaS is that it hides the complexity of managing parallelism, which is mandatory for "right-time" BI. Indeed, systems like MapReduce were specifically conceived for analytical tasks.

8.4 Software as a Service (SaaS)

As defined in [29], SaaS allows to run the providers application. The consumer does not manage or control the underlying infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. Software services allow a consumer to select a specific software instance without the need to be aware of where and how it is hosted. For example, a developer can request a word processor instance (i.e., Google apps) without having to be aware of which OS or hardware the application will run on. This allows the consumer to focus on the characteristics of the application and gives to the provider the freedom to fulfill the request with any resources that meet the need.

These are services that provide remote (e.g., Web-based) access to software that is actually running in provider's machines. As pointed out in [5], both (i.e., consumers as well as providers) benefit from this architecture. On the one hand, consumers gain access anytime and anywhere, and can easily share data and collaborate. This can bring huge benefits from the BI perspective, as data analysis could be carried out or studied on-the-fly with portable devices such as smart phones. Moreover, the pay-as-you-go model is also more beneficial than licensing where you pay per number of CPUs or users, for example. You only pay for the software if you really need to analyze your data. On the other hand, providers simplify software installation and maintenance, and centralized version control.

As explained in [20], the percentage of today revenue from BI as a service offerings is insignificant compared to the overall BI platforms market (less than 5%). [20] only mentions SAP/Business Objects "On Demand" as a product with a significant presence. Nevertheless, the penetration in the market depends on the specific kind of applications, and some are more mature than others. Gartner's report shows that SaaS versions of ERP (Enterprise Resource Management) are much less popular (below 1% of all ERP) than those of CRM (Customer Relationship Management, e.g., *Salesforce.com* or *Cloud9 Analytics*) or SCM (Supply Chain Management, e.g., *Oco*) systems (around 12% and 18% respectively). Examples of other BI software offered as a service are reporting, dashboarding, etc. (e.g., *Microstrategy*, *Quantivo*, *Panorama* or *ColdLight Neuron*). Let us analyze the former in the following sections.

8.4.1 Supply Chain Management (SCM)

As explained in [25], SCM systems enable the firm to model its existing supply chain, generate demand forecast for products, and develop optimal sourcing and manufacturing plans. They allow to exchange and share information between partners (i.e., producers and consumers) in the supply chain. SCM allows to minimize the *Bullwhip effect*, which is the distortion of information about the demand for a product as it passes from one entity to the next across the supply chain (i.e., little changes in the demand of the final consumer generate huge variations in the raw materials).

In the beginning, these systems where developed for each company. Afterwards, Electronic Data Interchange (EDI) appeared and allowed to efficiently exchange documents like orders or bills. However they yet did not help to manage the process. Nowadays, SCM systems allow a pull-based (i.e. build-to-order) model, where the customer order triggers events in the supply chain, so that production, inventory and shipping can be easily adjusted to real demand. In the future, the Internet could allow that all companies in the supply chain can see what others (not only immediate providers and consumers in the chain) need and generate at any time. This would facilitate not only fast response but also more precisely forecasting consumption.

However, this is not easy to achieve, because requires several organizations to share information which is an organizational, as well as a technological challenge. From the technological point of view, services (and service architectures) can be really helpful at this point.

8.4.2 Customer Relationship Management (CRM)

The concept of CRM appeared with this century. It can be defined as having an integral view of each and every client in real time. The goal is to be able to make fast, informed decisions regarding marketing strategies or competitive positioning (i.e., differentiate your offering and create value for your customers). This is achieved by tightening seller-buyer relationships.

Despite CRM has a big technological component, it is usually seen as a way of doing things in the company. As explained in [25], a CRM system captures and integrates customer data from all over the organization, consolidates and analyzes it, and distributes the results to various systems and customer touch points across the enterprise. Nevertheless, hardware and software must be accompanied by the definition of the corresponding processes or workflows, objectives and indicators.

To achieve these goals, the whole company must be involved in the deployment of the CRM. If such change generates more costs than benefits (specially to the marketing department), it will hardly succeed. Some web-based CRM software systems exist that can be licensed for a few hundred euros per user per month. In front of this, buying, installing and personalizing the technology can cost millions of euros to the company. Thus, Cloud Computing and

a service approach makes feasible for small companies to implement a CRM, or at least try to do it reducing the risk in case of failure of the project.

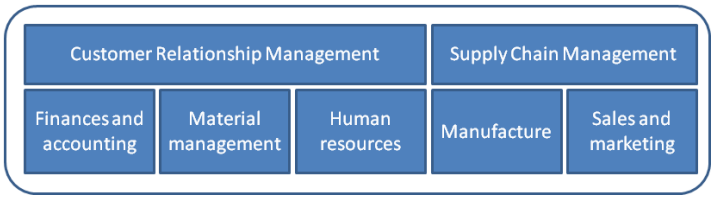

Fig. 8.6. Typical ERP modules

8.4.3 Enterprise Resource Planning (ERP)

Historically, enterprises have deployed independent systems to manage different processes in the company, which leads to, not only separate software and hardware, but also separate data that eventually have to be merged and integrated to make decisions. ERP systems try to solve this by providing one common repository where data are collected from all business processes. As depicted in Figure 8.6, different modules are available implementing common processes in the enterprises (e.g., Finances and accounting, Material management, Human resources, Manufacturing, Sales and marketing, etc.), and they just have to be properly configured and tuned up to behave as close as possible to the way a specific company runs the business. This helps not only the business processes themselves, since they share the information, but also facilitates decision making (most ERP systems have SCM and CRM modules).

Acquiring a software package always makes your company dependant on the software provider (obviously for installation, but also for upgrading and maintenance), and switching from one provider to another may not be easy. This is specially true in the case of ERP, because such systems drive all the operation of the company. Indeed, things are even worse if the ERP is contracted as a service, because the provider also hosts the data (special care must be taken at this point to guarantee that we may move our data out of the system if so desired). Upgrading and maintenance are much easier (and probably cheaper), but security issues pop up. SAP Business Suite, Oracle's e-Business Suite, Microsoft's Dynamics suite, and IBM's WebSphere are examples (and use Web services and SOA inside).

8.5 Business Process as a Service (BaaS)

Nowadays, the rapid changes in society, in general, and business in particular make enterprises an ever-evolving entity. Accordingly, applications and information systems are required to evolve quicker than ever and match the organizations needs. Special techniques are necessary to support this, allowing IT

to evolve at the same pace as business requires. One promising approach is to divide the processes in our business into units of processing logic. These units are collections of units of work that can be provided as a *service* (either from inside or outside the company). A service hence provides a specific capacity for the user and ideally, these units should be highly both customizable and reusable in such a way that complex systems could be seen as a myriad of them interacting as a whole.

A promising paradigm to tackle this scenario are Service-Based Applications (SBAs), see [34]. Software services constitute self-contained computational elements that support flexible composition of loosely coupled distributed software systems. This implies fundamental changes to how software is developed, deployed and maintained. The relevance of software services is increasing given the crucial role played by the Internet: on the one hand data distribution is a fact for most real organizations (either naturally distributed by geographic criteria or artificially like in a cloud to exploit parallelism). On the other hand, global connectivity brings services to another level, as they can be thought as pervasive, ubiquitous and highly dynamic units which could be searched, browsed and listed. Indeed, a clear example of SBAs can be found in the current *App stores* available for smart phones and tablets, which are nowadays already available for browsers and, in turn, for laptops and desktop computers.

Indeed, nothing prevents us to generalize the concept of SBAs not only to applications but to whole information systems inter and intra-organizations. According to the NIST Working Group, BaaS focuses on providing existing business processes through a cloud. If there is an existing process whose specifications are known, it can be provided as a service within the catalog. This allows the service provider to automate any steps within the process while leaving the changes transparent to the service consumer. There already are some examples in this direction, and it is rather usual to find outsourced paying methods in the Web (e.g., PayPal or Moneybookers) or geolocation stuff (e.g., by combining the smart phones built-in GPS features plus Google Maps).

BaaS, however, introduces new challenges and specific techniques to engineer service-based systems (e.g., Service Oriented Architectures and Business Process Modeling) have emerged lately. According to [34], we can distinguish between challenges related to service technologies (i.e., the means) and service principles, techniques and methods to develop service-based systems. The main challenges regarding service technologies are the following:

Service Infrastructure: It refers to basic infrastructure on top of which the service-based systems are built. It refers to the most basic service layer responsible for communication primitives and patterns, architectural constructs to connect heterogeneous systems, service access and a runtime environment for services execution. It is also responsible for service search, identification and publication. In the context of this paper, the first two previously discussed service layers (i.e., IaaS and PaaS) could sit below

BaaS and provide the needed service infrastructure to built on its top service-based systems.

Service Composition and Coordination: Service oriented systems loosely couple logic units (i.e., services) to aggregate them in an interoperable way and produce a single, complex service. SBAs are built on Service Oriented Architectures (SOA). An important property of services taking part in an SOA is their composability (see [10]). There are several options to service composition such as service orchestration, or service choreography among others (see Section 8.5.1).

Business Process Management (BPM): BPM can be defined as mechanisms to understand, capture, represent and manage organizations, and is aimed at developing end-to-end business processes. Its main objective is to fill the gap between business processes and IT applications, to better understand an application's requirements. It can be understood as a management principle, but also represents suites of software aimed at bridging IT systems and people (see [17]). Current SBAs typically involve well-defined processes (such as payment processing, shipping, tracking, etc.) which must be understood as a whole. However, managing a finer-granularity (e.g., business data, events, Quality of Service -QoS- agreement, Key Performance Indicators -KPIs-, etc.) is needed to guarantee a cohesive and continuous information flow, at the level of business design and not merely at technical level. Business Rules Management Systems (e.g., Drools or IBM WebSphere ILOG JRules) are a natural partner for BPM systems (like IBM BPM tool), as they allow to store the business decision logic in a centralized repository.

On the other hand, on top of service technologies, service principles, techniques and methods focus on developing service-based systems. Basically, [34] highlights the relevance to come up with a specific lifecycle for service oriented systems. Indeed, it cannot be assumed that existing applications can be imported into an SOA by merely creating wrappers and leaving the underlying component untouched. A service-based development method is required to specify, construct refine and customize highly flexible service-based systems. For example, the adapting and evolution of services based on monitoring performance indicators and an automatic Quality of Service (QoS) negotiation between services are reasons why the life-cycle must be extended. Thus, service-based systems are not just a thin SOAP/WSDL/UDDI layer on top of existing systems or components. In short, new challenges (regarding traditional systems) about developing service-based systems can be summarized as follows:

Service Engineering and Design: This should provide specifications for composition and coordination of the services forming the deployed system, which guarantee it does not produce spurious results. The process should lead to high-quality service-based systems. The BPM layer would also be included in this item.

Service Quality: Service quality definition, negotiation and assurance (ideally in an automatic way) is a key aspect for service-based systems at all levels. This includes assessing a service quality level both for controlling the agreed communication quality level between services forming the system and for agreeing on the quality of the output generated. Relevantly, end-to-end quality provision implies to propagate quality control on attributes from the service infrastructure, service composition and business process management. Therefore, it is transversal to the whole system.

Service Adaptation and Monitoring: Tightly related to the previous item, service-based systems should adapt and evolve to answer contextual changes and even predict upcoming problems. In this sense, monitoring, predicting and governing the activities of the service-based system emerge as key aspects.

For these reasons, as discussed, an entire specific service lifecycle (identifying, finding, designing, developing, deploying, evolving, quality assuring and maintaining services) must be considered. In the remainder of this section, we will focus on two specific concepts related to BaaS from which BI can vastly benefit. Firstly, we analyze how SOA can be used as an architecture for providing BI, and then how to define, negotiate and assure the quality of the BI services.

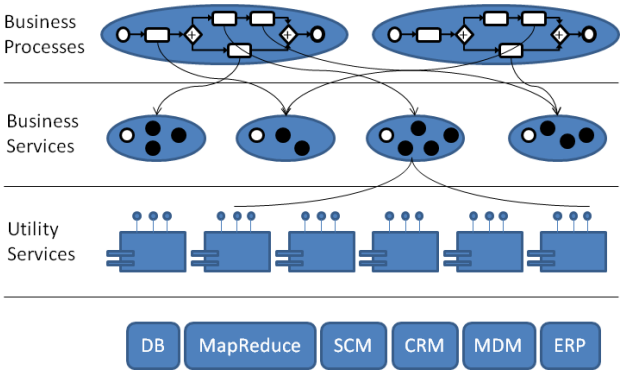

Fig. 8.7. Service Oriented Architecture

8.5.1 Service Oriented Architecture (SOA)

As defined in [15], SOA is a term that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic, namely processes or activities (see Figure 8.7). Individually, these units can be distributed. Web services, including standards such as WSDL (Web Services Description

Language), SOAP (Simple Object Access Protocol) and WS-BPEL (Web Services - Business Process Execution Language) are the most popular realization of SOA (see [10]).

Three overlapping kinds of services exist:

Business service: It represents the most fundamental business block and encapsulates a distinct set of business logic. At this level, Business Process Modelling (BPM) is a natural complement to SOA towards the objective of aligning technical initiatives with strategic goals of the business (see [33]).

Utility service: This provides generic services designed for potential reuse (black circles in Figure 8.7).

Controller service: It is responsible to coordinate service-to-service composition members (white circles in Figure 8.7).

Compared to two-tier client-server architecture, SOA offers more than one decomposition layer (i.e., a service can be successively decomposed into other smaller services). Moreover, processing units are much smaller than typical monolithic client or server applications. Multi-tiered client-server applications do not necessarily follow SOA, even if they are distributed and incorporate Internet technology, among other reasons, because the components are still tightly coupled. Even using Web Services in these applications does not entail SOA, because they do not guarantee the independence between components.

Thus, eight common principles to SOA are stated in [15]:

Reusability: Services are designed to support potential reuse.

Loose coupling: Services must be designed to interact without the need for tight, cross-service dependencies.

Contract: They need not to share anything but a formal contract that describes each service and defines the terms of information exchange.

Abstraction: The only part of a service that is visible to the outside world is what is exposed via service contract.

Composability: Services may compose other services.

Autonomy: The service does not depend on other services for it to execute its governance.

Statelessness: Services should be designed to maximize statelessness even if that means deferring state management elsewhere.

Discoverability: Services should allow their descriptions to be discovered and understood by human as well as service requestors that may be able to make use of their logic.

Each one of these characteristics emphasizes one different aspect of services. However, all of them pivot around the concept of reusing software as much as possible. Thus, in order to reuse, we must also be able to compose different services. The interaction of a group of services working together to complete a task can be referred to as a *service activity*. This, as explained in [15], can also be achieved in some different ways:

Primitive activity implements synchronous communication between two services. It is short-lived, and typically involves only one message exchange between them.

Complex activity involves many services and messages exchanged among them, that allow to carry out complex tasks. It usually spans a longer period of time and requires some coordination framework.

- Atomic Transaction wraps a series of changes into a single action (usually ACID-compliant). Its internals provide some protocols to guarantee it (e.g., two-phase commit). Atomic transactions play an important role in ensuring service quality and lead to a robust execution environment.
- Business Activity participants are required to follow specific rules defined by protocols. Oppositely to an atomic transaction, it spans hours, days or even weeks. Moreover, business activities provide an optional compensation process that can be invoked when exception conditions are encountered (i.e., they are not expected to rollback any changes). An added difficulty for this kind of activities is that they cannot expect to retain the participants for the duration of the activity, since the coordinator does not have any control over them.
  - Orchestration participants interact with a central engine, which provides the workflow logic. This logic can be expressed, for example, in BPMN (Business Process Modeling Notation) and BPEL (Business Process Execution Language),see [49].
  - Choreography is intended for public message exchanges, so that collaboration between services can be implemented without needing the central coordinator of the workflow logic. This is achieved by assigning roles to the participants. Choreographies are conceived to achieve collaboration among services from different, independent providers.

Using BPM to model the whole BI process definitely makes sense, since BI processes are completely business-oriented and can be modeled as workflows (specifically, as data transformation workflows). However, it seems reasonable to only outsource (i.e., contracting external services to carry out the task) certain BI tasks within BI processes. For example, contracting additional services providing input data for the BI process (e.g., stock markets data or news feeders) or for providing external rules used in the process to transform data (e.g., spam detecting and mail filters for data cleansing or external dictionaries / ontologies to support data integration) fully make sense, but it would not be reasonable to outsource intermediate tasks within a business process, as it would provoke massive data shipping between services (as discussed, Cloud Computing entails that we are charged according to the amount of data shipped). For example, we may contract input data (such as the stock market daily data), load it into a service-based system and carry out as many transformations as needed to match up the BI process goals. However, shipping data to external (outsourced) intermediate services in the middle of the process and send it back after some transformations is

not reasonable cost-wise. Following this idea, any BI process can be seen as data transformation from the data sources to the Key Performance Indicators (KPI) exploited by the managers. Data manipulated all over the process is transformed in two different ways: (i) dynamic on-line transformations (e.g., OLAP engines), which allow the users to query sources (such as decisional databases) and produce data analysis on-the-fly, and (ii) static off-line transformations, which would match the traditional ETL process to produce the sources queried dynamically and on-line by the user (i.e., to populate the decisional database). Given this classification, SOA seems more promising for the latter, as discussed below.

In [47], the framework offers an extensible palette of template activities for ETL processing. These activities are defined there as an amount of work which is processed by a combination of resource and computer applications. Moreover, this activity can be performed in a black-box manner. Note that the same word "activity" is also used in [15] to refer to sets of coordinated services. Thus, each ETL flow can be considered as a separate business process. Then, each business process would be implemented as a complex activity. Some would be atomic transactions able to rollback to a safe state. However, most of them would be business activities, demanding the definition of a compensation process to leave data consistent in case of a not completely successful execution. These activities would be defined in the form of orchestrations. There are works (for example, [13]) that already propose the usage of an SOA architecture, where different data mining elements interact to complete the analysis of data (financial market in this case). The characteristics of services would allow to reuse components or just change the provider (if we find another one improving the quality of the BI process).

As explained in [8], we can model the ETL workflow as a business process. This would allow to have a conceptual view that can be mapped later to the implementation. It would hide low-level IT events resulting to be more comprehensible to analysts. In some cases this would result really helpful for business managers to, not only understand, but also define how data have to be transformed.

We find in [34] a list of major challenges for the near future of services. Among them, we find some that would clearly be useful to BI, like "Infrastructure support for data integration" and "Semantically enhanced service discovery". On the other hand, it also outlines the current lack of tools for supporting the evolution and adaptation of [business] processes, so that it is hard to define compositions of distributed [business] processes that work properly under all circumstances.

8.5.2 Quality of Service (QoS): Definition, Negotiation and Assurance

With services, quality "specifications" come from multiple simultaneous sources, including the company and the individual customers. The company presents specifications as standard operating procedures. The customer

presents specifications based on their need-driven expectations for changes to their process-inputs. Misalignment between company- and customer-specifications for the service process leads to dissatisfaction, even when the process goes exactly as it was designed. The misalignment of specifications can be avoided through communication. However, if the service performance does not address individual customer needs, the customer will not require the service. With services, expectations are often subjectively acquired and subjectively defined. Therefore service providers need to be careful when attempting to define expectations for customers. Too low expectations can lead to loss of sales. Too high expectations can lead to disappointment and lost future sales [40].

With this spirit, service-based systems must negotiate and agree quality aspects regarding the system. Service providers need to characterize their services to define both the offered functionalities and the offered quality. For example, according to [37], these quality aspects may embrace quality attributes regarding runtime, transaction, configuration management and security features. Furthermore, not only the need to define, negotiate and agree the QoS is growing, but also its eventual validation and verification at run-time.

As stated in [34], QoS is measured by the degree to which applications, systems, networks, and all other elements of the IT infrastructure support availability of services at a required level of quality under all access and load conditions. As discussed in [44], measuring and evaluating performance remains a difficult problem. A service should be developed and delivered to achieve maximum customer satisfaction at minimum cost. To facilitate this task, they provide a tentative list of measures partitioned in four categories to evaluate the service:

Categories	**Example Evaluation Measures**
Input	Demand
	Supply
	Cost
Process	Performance (Quality, Reliability, Speed, Throughput)
	Satisfaction (Efficiency, Effectiveness)
	Safeguards (Privacy, Security, Safety)
Outcome	Customization
	Satisfaction
	Convenience (Availability, Accessibility)
	Robustness (Comprehensiveness, Adaptability, Flexibility)
Systemic	Consistency
	Equity
	Reproducibility
	Sustainability

The input and process measures serve to explain the resultant outcome. Input measures indicate the potential of our system and what it needs to run (i.e., they measure how the input looks like; e.g., demanded input or

cost), whereas process measures (e.g., speed, efficiency, privacy, etc.) consider how the activity is performed but not the impact it has. This is the main reason for the third set of measures, which focuses on measuring the ultimate results (e.g., its degree of customization or the user satisfaction). Finally, the systemic measures are regarded as impact measures over other systems or services (e.g., sustainability or consistency).

As shown in [14], data intensive applications can benefit from an agreement in the QoS in advance to the actual data retrieval. A Service Level Agreement (SLA) is based on a set of measurable characteristics of a service known as Service Level Objectives (SLO) (e.g., price, availability, response time, number of available tuples, number of retrieved tuples, query cost, locality and legal issues, etc.). Note that SLOs of complex activities must be decomposed for the participants in the activities. The other way round, estimation of each characteristic must be aggregated for the services composing such activities.

As pointed out in [8], implementing the ETL workflow as a business process would also help to manage its monitoring and reporting. In this sense, another important challenge is the implementation of "QoS-aware service compositions" that takes into account the performed data transformations (see [27]).

8.6 Discussion

This section contains an overall discussion of advantages and disadvantages of using services (at any of the four above mentioned levels) for BI. First of all, one key aspect regarding a service is that it entails outsourcing (in general, from the company, but also just from the department or business unit). Roughly speaking, it allows the company to spend the efforts in making decisions instead of building decisional systems. Before other considerations, thus, we must take into account that, as pointed out in [18], outsourcing has benefits and risks from the customer point of view. The list of these benefits can be summarized as follows:

- Allows the firm to focus on its core competence.
- Decreases cost by purchasing from an outside source rather than performing in-house.
- Provides access to latest technology without investment.
- Leverages benefits from a supplier who has economies of scale.

Oppositely, main risks to consider are:

- Loss of direct control over quality.
- Exposure to data security issues.
- Dependence on one supplier compromises future negotiation leverage.
- Additional coordination expense and delays.
- Atrophy of in-house capacity to perform outsourced services.

Thus, the key question is how to guarantee that the selected service does really match our needs. In other words, how to minimize the outsourcing risks and maximize its benefits. In [18], the authors highlight some criteria that might be used to select a service provider:

Convenience: Some locality issues may be considered due to legal conditions or connectivity.
Dependability: Is the provider reliable?
Personalization: Customization is an important, not to say crucial, characteristic in any service, as it must adapt to our needs.
Price: Obviously, the cheaper (provided a given set of features), the better.
Quality: Defined as a function of previous expectations and perception of the experience.
Reputation: Previous customers' experiences are important.
Speed: How long it will take to complete the process.
Security: In the broad sense of CIA (i.e., Confidentiality, Integrity and Availability), as pointed out in ISO/IEC 17799, is a must.

All these criteria happen to be yet relevant for BI but, specifically, security, quality, and personalization shall draw the attention of BI customers. Indeed, regarding BI, security is a major concern for them, since the (potentially sensible) data of the company has to be transferred, managed and even stored by service providers, raising, consequently some inherent questions such as: Are my data safe? Will data be available when needed? Moreover, the provider must guarantee that data are not only geographically distributed, but also replicated to reduce their vulnerability to catastrophes and network partitions. Regarding quality, BI processes should guarantee a minimum threshold of quality previously negotiated and agreed with the provider. Furthermore, monitoring services to check whether the agreed quality level is reached or not should be available for the user. Finally, how to customize, adapt and evolve the available services to fully match customers' specific analytical needs is a challenge for BI providers, since by definition, analysis requirements will be different for each company.

On the other hand, the advantage of externalizing is that you pay for what you really need at every moment. This solves the problem of underusing the customer's data center, but from the provider's viewpoint, a capacity management problem appears to deal with variations in the demand. Given that analytical tasks are data as well as computation intensive, they can present special problems to absorb their load without violating the service contract.

Nevertheless, you should note that the benefit at the customer size is huge, because (if using a scalable platform) the result of a computation can be obtained in as few time as desired (by just hiring more parallel resources) without a substantial increment in the overall cost. The problem the customer has to face is that of transferring data to the providers side (if they were not already there). Internet transfers are slow and data movements in and out of

a provider are usually charged. This can make the difference given the huge amount of data in BI applications.

Leaving aside the economical issue of moving data in and out of the provider's machines, a service architecture allows to face one of the most challenging BI problems we have today, i.e., the elastic capacity provides the flexibility to analyze in "right-time" the huge amount of unstructured data flowing through the Internet (e.g., e-mails, twits, reviews, comments, evaluations, etc.). The challenge is triple: we firstly have to deal with a humongous amount of data; these data are not structured; and finally, we want to make a decision as soon as possible to avoid missing a business opportunity.

Also from the customer point of view, by hosting on one machine the BI applications used by many companies, licenses for background software only have to be paid once, and the software code updates are immediately available for all users, who have access to their data anytime and anywhere. However, this raises the problem of lack of dedicated IT personnel (i.e., atrophy of in-house capacity) and loss of access to the backstage. The end user should be self-sufficient. Some recent works go in this direction, either helping on the management of the operating system [22], the DBMS [7] with a DW workload [16], the ETL processes [8], the multidimensional design [38, 39] or the queries posed [19, 24].

8.7 Business Intelligence on Services

In the previous sections we have discussed how BI can benefit from service science, whereas in this section we focus on the other way round: how services can benefit from BI characteristics and techniques and how to apply them to improve service-oriented systems.

It has already been shown how BI might help to turn traditional manufacturing into services. Basically, service principles (such as customization, adaptiveness, quality negotiation, etc.) can be applied by exploring the available data of the organization and extract relevant knowledge, which will allow us to, for example, adapt our products to customers. To undertake this process, BI techniques are of great relevance. As pointed out in [18], in contrast to manufacturers, knowing the relationships with customers is a significant competitive advantage in service companies (since they are co-producers of the services). Thus, CRM (as explained in Section 8.4) is really important in this sense to the point of even becoming a barrier to entry for competitors. These systems are used to, based on how profitable their business is, code customers with instructions for service staff, route them in call centers to place them in different queues, target the offers and share their data with other firms. The problem here is that we must track all details about customers, resulting in a huge volume of data, but again, this is the main objective of traditional BI systems such as Geographic Information Systems (GIS) or data warehousing.

Relevantly, by allowing customization of their products manufacturers gain service characteristics. In this case, it is specially important to coordinate the whole chain from parts suppliers to retailers. Thus, SCM (see Section 8.4) represents a competitive advantage as well as a challenge because of the network model we have in general, and the level of uncertainty.

Furthermore, making decisions to improve service quality is also a must. Quality is a differentiation strategy to gain competitive advantage. Nevertheless, as stated in [40], it tends to be subjective and difficult to scale, because it is defined based on expectations and perception of the service. Nevertheless, this difficulty does not impede that KPIs are usually defined for the different processes of the companies.

All in all, information, understood as the result of processing, manipulating and organizing data in a way that adds new knowledge to the organization receiving it, is the keystone of service-based systems. Consequently, databases are a great asset to service companies and, for example, they can sell information about their customers (if legality allows it) or can directly use it for marketing purposes. At this point, traditional BI techniques such as reporting, OLAP and data mining play a critical role. In the following, we will focus on how services monitoring can be taken to another level by applying BI techniques and exploiting the derived knowledge for assuring the quality of services, allowing them to evolve and adapt and, in short, make them perform more efficiently.

8.7.1 Monitoring Our Services

Monitoring is a key aspect to guarantee the success of service-based systems. Monitoring refers to the need to monitor the business processes in order to (re)-design, adapt and evolve them (see [41]). Tightly related to BPM, an example of software covering this area is IBM Business Monitor. Most companies already have a tool (commonly known as "Balance scorecard") that allows to follow the consequences of the decisions made and easily check whether they succeed or fail. The main idea behind monitoring is that you cannot control what you cannot measure. Therefore, relevant knowledge must be available in order to make better decisions. Traditionally, a set of KPIs are defined and showed in a graphical way (e.g., balance scorecard, digital dashboards or any other Performance Measurements System). To do so, normally data are first gathered and integrated in decisional databases which are later used to nurture dashboards, balance scorecards, etc.

In its broader sense, monitoring has successfully been applied in BI for a while (see [8]). The innovative aspect in service-oriented systems, is that Key Performance Indicators (KPIs) are tightly related to business processes from the very beginning at the definition and design steps. Indeed, managers and decision makers could define them in high level languages such as BPEL and BPMN and make them flow all over the technical aspects bridging the gap. As an example following this principle, Figure 8.8 presents IBM's Websphere

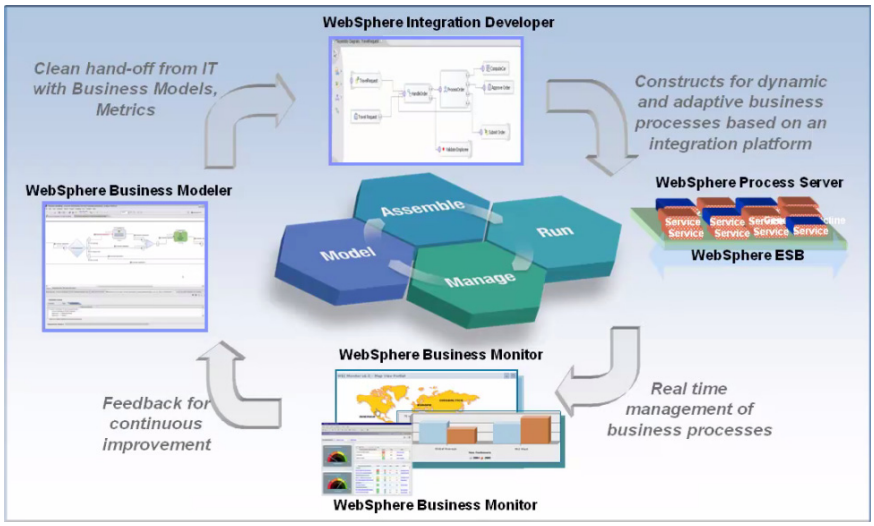

Fig. 8.8. IBM WebSphere BPM components

components, which follow this principle to let the system evolve and adapt to the customer needs. In this framework, the manager would firstly define the process using a BPMN diagram. Then, this is automatically passed to the developers that have to implement and assemble the corresponding service components. Those components are executed in a runtime environment, than can be interactively monitored and modified.

However, KPIs cannot be only conceived as performance indicators at the client end but also at the provider end. Indeed, they can also be internally applied to measure the quality of services. For example, we can define KPIs to measure if the Service Level Objectives (SLOs) are reached (and thus, if the provider - customer agreement is fulfilled). These KPIs do not directly impact on the business processes implemented but on the quality of the selected services. This valuable knowledge can be further exploited to adapt and evolve our SLOs and eventually, our service agreements. In this sense, [8] automates the design, implementation and adaption of ETL for business processes, by recording and monitoring their events. It should be during the definition of this ETL process that the KPIs of the company are defined. Indeed, in this work, the authors also explain how the definition of KPIs is also facilitated in their framework.

As pointed out in [45], it is really interesting to establish the correlation between IT performance and business performance metrics (i.e., KPIs). Thus, metrics (beyond failure, availability and response time) should be defined to data processing and see how these affect the business. Importantly, many data are generated in BPM systems. [31] classifies these data and analyzes

their complexity in terms of volume and variance. Obviously, the more data we have and more they vary, the harder it will be to analyze them. At this point classical BI techniques such as reporting, OLAP and data mining become essential to enable successful data analysis. Reporting is of great value to periodically trigger well-known analytical processes to control workflow processes, whereas OLAP and data mining provide means to facilitate advanced analytical tasks based on KPIs. OLAP principles allow us to easily aggregate performance indicators and analyze their values according to different perspectives of analysis, whereas data mining provides algorithms to derive and infer hidden knowledge based on statistical data transformations. All in all, BI techniques become essential for a successful analysis of KPIs, both at customer and provider end.

8.8 Conclusions

Firstly, we have carefully analyzed the specific characteristics of services. Then, we have studied how Business Intelligence can benefit from services at four different levels (i.e. IaaS, PaaS, SaaS and BaaS):

- In the first case, the power of Cloud Computing can help the costly BI applications.
- In the second, specific platforms (e.g., BigTable and MapReduce) have been developed to benefit from world scale parallelism in analytical tasks.
- In the third, market studies show the relevance of BI software as a service in the overall software as a service market.
- Finally, BPM and ETL modeling are clearly linked as part of a global design process.

On the other hand, we also outlined the importance of monitoring service business and showed how this can be done.

Acknowledgements. We would like to thank Ferran Sabaté for his help with CRM and SCM, as well as Victor López and Oscar Fernández for their insights and help with IBM products. Also a special acknowledge to the reviewers of the first version of the paper, for their help to improve the structure and readability of the paper.

This work has been partly supported by the Spanish Ministerio de Ciencia e Innovación under project TIN2008-03863.

References

1. Abadi, D.J.: Data management in the cloud: Limitations and opportunities. IEEE Data Engineering Bulletin 32(1), 3–12 (2009)
2. Aboulnaga, A., Salem, K., Soror, A.A., Minhas, U.F., Kokosielis, P., Kamath, S.: Deploying database appliances in the cloud. IEEE Data Eng. Bull. 32(1), 13–20 (2009)

3. Apache: Hadoop, http://hadoopapache.org/
4. Apache: HBase, http://hbaseapache.org/
5. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Commun. ACM 53(4), 50–58 (2010)
6. Battré, D., Ewen, S., Hueske, F., Kao, O., Markl, V., Warneke, D.: Nephele/PACTs: A programming model and execution framework for web-scale analytical processing. In: Proceedings of the 1st ACM Symposium on Cloud computing (SoCC), pp. 119–130. ACM (2010)
7. Bowman, I.T., Bumbulis, P., Farrar, D., Goel, A.K., Lucier, B., Nica, A., Paulley, G.N., Smirnios, J., Young-Lai, M.: SQL Anywhere: An Embeddable DBMS. IEEE Data Engineering Bulletin 30(3), 29–36 (2007)
8. Castellanos, M., Simitsis, A., Wilkinson, K., Dayal, U.: Automating the loading of business process data warehouses. In: 12th International Conference on Extending Database Technology (EDBT), pp. 612–623. ACM (2009)
9. Chang, F., et al.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems (TOCS) 26(2) (2008); preliminary version published in OSDI 2006
10. Curbera, F., Duftler, M.J., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6(2), 86–93 (2002)
11. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: 6th Symposium on Operating Systems Design and Implementation (OSDI), pp. 137–150 (2004)
12. DeWitt, D.J., Gray, J.: Parallel database systems: The future of high performance database systems. Commun. ACM 35(6), 85–98 (1992)
13. Díaz, D., Zaki, M., Theodoulidis, B., Sampaio, P.: A Systematic Framework for the Analysis and Development of Financial Market Monitoring Systems. In: SRII Global Conference (2011)
14. Engelbrecht, G., Bisbal, J., Benkner, S., Frangi, A.F.: Towards negotiable SLA-based QoS Support for Data Services. In: International Conference on Grid Computing (Grid), pp. 259–265 (2010)
15. Erl, T.: Service Oriented Architecture. Prentice Hall (2006)
16. Favre, C., Bentayeb, F., Boussaid, O.: Evolution of Data Warehouses' Optimization: A Workload Perspective. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 13–22. Springer, Heidelberg (2007)
17. Ferguson, D.F., Stockton, M.L.: Enterprise Business Process Management - Architecture, Technology and Standards. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 1–15. Springer, Heidelberg (2006)
18. Fitzsimmons, J., Fitzsimmons, M.: Service Management, 6th edn. McGraw-Hill (2008)
19. Giacometti, A., Marcel, P., Negre, E., Soulet, A.: Query Recommendations for OLAP Discovery-Driven Analysis. IJDWM 7(2), 1–25 (2011)
20. Hostmann, B.: Business Intelligence as a Service: Findings and Recomendations. Research G00164653, Gartner (January 2009)
21. Inmon, W., Strauss, D., Neushloss, G.: DW 2.0. The Architecture for the Next Generation of Data Warehousing. Morgan Kaufmann (2008)
22. Isard, M.: Autopilot: automatic data center management. Operating Systems Review 41(2), 60–67 (2007)

23. Jiang, D., Ooi, B.C., Shi, L., Wu, S.: The performance of mapreduce: An in-depth study. PVLDB 3(1), 472–483 (2010)
24. Koutrika, G., Ioannidis, Y.E.: Personalization of queries in database systems. In: Proceedings of the 20th International Conference on Data Engineering (ICDE), pp. 597–608. IEEE Computer Society (2004)
25. Laudon, K.C., Laudon, J.P.: Management Information Systems: managing the digital firm. Prentice-Hall (2010)
26. Lovelock, C., Wright, L.: Services Marketing: People, Technology, Strategy, 6th edn. Prentice-Hall (2007)
27. Marotta, A., Piedrabuena, F., Abelló, A.: Managing Quality Properties in a ROLAP Environment. In: Martinez, F.H., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 127–141. Springer, Heidelberg (2006)
28. Meijer, E., Bierman, G.: A Co-Relational Model of Data for Large Shared Data Banks. Communication of the ACM 54(4) (2011)
29. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Special Publication 800-145, National Institute of Standards and Technology (January 2011), draft
30. Melnik, S., et al.: Dremel: Interactive Analysis of Web-Scale Datasets. Proceedings of the VLDB Endowment (PVLDB) 3(1), 330–339 (2010)
31. zur Muehlen, M.: Volume versus variance: Implications of data-intensive workflows. IEEE Data Eng. Bull. 32(3), 42–47 (2009)
32. Özsu, T., Valduriez, P.: Principles of distributed database systems, 3rd edn. Prentice-Hall, Inc. (2011)
33. Papazoglou, M.P.: Web Services: Principles and Technology. Prentice Hall (2007)
34. Papazoglou, M., et al. (eds.): Service Research Challenges and Solutions for the Future Internet, vol. 6500. Springer, Heidelberg (2010)
35. Pavlo, A.: et al.: A comparison of approaches to large-scale data analysis. In: SIGMOD Int. Conf. on Management of Data, pp. 165–178. ACM (2009)
36. Pedersen, T.B.: Research challenges for cloud intelligence: invited talk. In: Proceedings of the 2010 EDBT/ICDT Workshops. ACM International Conference Proceeding Series (2010)
37. Ran, S.: A model for web services discovery with qos. SIGecom Exchanges 4(1), 1–10 (2003)
38. Romero, O., Abelló, A.: Automatic validation of requirements to support multidimensional design. Data Knowledge Engineering 69(9), 917–942 (2010)
39. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data Knowledge Engineering 69(11), 1138–1157 (2010)
40. Sampson, S.E.: Understanding Service Businesses: Applying Principles of Unified Services Theory, 2nd edn. John Wiley& Sons (2001)
41. Sánchez, L., García, F., Ruiz, F., Piattini, M.: Measurement in Business Processes: A Systematic Review. Business Process Management Journal 16(1), 114–134 (2010)
42. Stonebraker, M.: et al.: C-Store: A Column-oriented DBMS. In: 31st Int. Conf. on Very Large Data Bases (VLDB), pp. 553–564. ACM (2005)
43. Stonebraker, M., et al.: MapReduce and parallel DBMSs: friends or foes? Communication of ACM 53(1), 64–71 (2010)
44. Tien, J., Berg, D.: A case for service systems engineering. Journal of Systems Science and Systems Engineering 12(1), 13–38 (2003)

45. Truong, H.L., Dustdar, S.: Integrating data for business process management. IEEE Data Eng. Bull. 32(3), 48–53 (2009)
46. Tsangaris, M.M., Kakaletris, G., Kllapi, H., Papanikos, G., Pentaris, F., Polydoras, P., Sitaridi, E., Stoumpos, V., Ioannidis, Y.E.: Dataflow processing and optimization on grid and cloud infrastructures. IEEE Data Eng. Bull. 32(1), 67–74 (2009)
47. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M., Skiadopoulos, S.: A generic and customizable framework for the design of ETL scenarios. Information Systems 30(7), 492–525 (2005)
48. Vogels, W.: A Head in the Cloud - The Power of Infrastructure as a Service. In: First workshop on Cloud Computing and Applications, CCA (2008)
49. Weske, M.: Business Process Management. Springer, Heidelberg (2007)
50. You, J., Xi, J., Zhang, C., Guo, G.: HDW: A High Performance Large Scale Data Warehouse. In: International Multi-Symposium of Computer and Computational Sciences (IMSCCS), pp. 200–202. IEEE (2008)