DIGITAL TECHNOLOGY
Handouts and Slides from lectures

2024 edition - in progress

B. Pernici (ed.)

Version 2.2 - March 19, 2024[1]
Modifications:
vers. 2.2 vs vers. 2.0x: minor edits in Ch. 1,2,3, Ch. 4 and 5 updated; new Ch. 6 (replaces slides
removed chapters on Data in social media and UN SDGs and data sources
Chapters after Ch. 6 are still under revision

Contents

Chapter 1

Introduction

1.1 Data and information

The term 'data' and the term 'information' are often considered as synonyms, however in information systems we need to distinguish between them and learn how to use these and related terms appropriately. In order to illustrate the differences, we will refer to the so-called pyramid of knowledge, also called the DIKW pyramid (Data. Information, Knowledge, Wisdom), shown in Figure 1.1.

It can be seen that data represents the basis of everything. In fact, the word *data* must be interpreted as the past participle of the verb "to give". With the term "given", we mean a fact, a measure, therefore an element that it models or describes a portion of the reality that is to be represented. Associated with a data item there is always its type which specifies the domain of values that such data item can assume. For example, if you want to represent the number of students enrolled in a class, the type of data must necessarily be numeric or, to be more precise, it must be a natural integer, since it is not correct to represent this portion of reality using negative numbers, much less using numbers with decimals. In addition to the type, a data item can be characterized by its unit of measurement. For example, if you want to represent the temperature in a classroom, in addition to associating a numeric type - this time with a real number - the measurement scale must also be indicated (for example, degrees Celsius or degrees Fahrenheit). Since the data collected can be numerous, the *database* discipline defines methods and tools to organize them within more or less complex and more or less flexible structures capable of making not only their storage easy and optimal but also their retrieval. In fact, the purpose of a database is to represent a portion of reality by storing a set of data that reflect this reality and make it available to users.

Following the scheme suggested by the pyramid of knowledge, the *information* is positioned above the data. As mentioned above, very often these two terms are often used interchangeably. In reality, information is built from data, and, in particular, information can be defined as the interpretation of a single

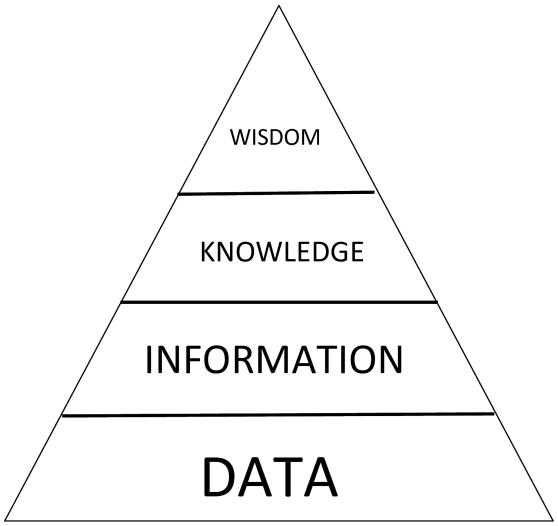


Figure 1.1: DIKW pyramid (source Wikipedia)

data item or a set of data. In fact, a data item taken individually, without any context, often remains useless. For example, having temperature data without knowing whether this temperature refers to a given internal environment, such as a given room, or the external environment in a given location does not make any sense. For this reason, for data to be useful, it must be combined with other data to define a context that allows us to better characterize the reality that is being represented. For example, it is helpful to associate a temperature with the date (and time) when it is measured and the room or location. For the number of students of a class, when it was computed and for which class. Information can therefore be defined as the output of queries addressed to a set of data (presumably organized in a database), such as "How many degrees are there today in the lecture hall B12.0.1?" Or "what is the average of students enrolled in a course with a given title in the last five years? ". Continuing to go up the pyramid, above the information is the *knowledge* which is obtainable by integrating information with experience. In fact, knowing that there are 32 degrees in the great conference hall in mid-December suggests that there is some anomaly as this value differs from what experience suggests, that is, from the average temperatures usually observed in that period. While information allows us to represent reality in a complete and useful way, knowledge guides any decisions that may affect the current situation. It should be emphasized how knowledge is acquired over time, through experience from one subject, and usually, it is difficult to transfer to other subjects. On the contrary, it is possible to transfer information or, at most, experience with the problem. For example, which temporal context should be considered for the calculation of the historical series with respect to the measurement of the observed temperature? Just the current day? A week, a month? What can be derived from it to support deci-

sions? For instance, is the room temperature always within acceptable ranges?

Finally, there is *wisdom* which represents an extension of the classic pyramid of knowledge (whose name suggests that at the top there is, in fact, knowledge) and which can be defined as the experience applied to knowledge to guide a subject to take the most suitable action at a given moment. Returning to the example, having understood - through the comparison between the current temperature of the auditorium and the historical temperature series - that there is an anomaly, can be a reason to turn off the heating as this action has an influence on the temperature.

The use of the pyramid as a geometric figure to represent these concepts is not accidental. First, it is layered, highlighting how upper levels are built from lower levels. Second, the area covered by each level decreases as the level increases. This wants to represent the level of *synthesis*: at the bottom, we have very numerous data and at a level of fine granularity, while on the opposite side, we have the wisdom represented with a smaller number of elements and therefore very synthetic and with coarse granularity.

From the point of view of data management in an IT system (i.e., the IT platform), as the level increases, the ability to manage the contents of the knowledge pyramid with automatic tools decreases. While the data is managed through database management systems (highly automated), the information can be described and abstracted to support decisions with automatic and semi-automatic tools, while knowledge presents aspects that only partially lend themselves to automated support.

1.2 Data in organisations

In organizations, **information** is used for communication, for process support and decision-making, for which information on possible strategic or control alternatives is needed. Information can also be, for some companies, a product. Information has the following characteristics that distinguish it from any other corporate resource: it is intangible, it is (infinitely) shareable (contrary to other resources, it is not destroyed nor consumed with use), self-regenerating (its use leads to the generation of new information). For example, in mobile communications, the information is used to obtain information on users, propose new rates, obtain feedback on marketing initiatives, and update rates. In a production line, it is necessary to store information about the jobs performed (for example, time required, percentage of damaged parts), identify problems, or improve the execution of the process. Managing information, therefore, translates into countless activities, which can be categorized into the following main ones:

- Create information.
- Get information.
- Process information.
- Store information.

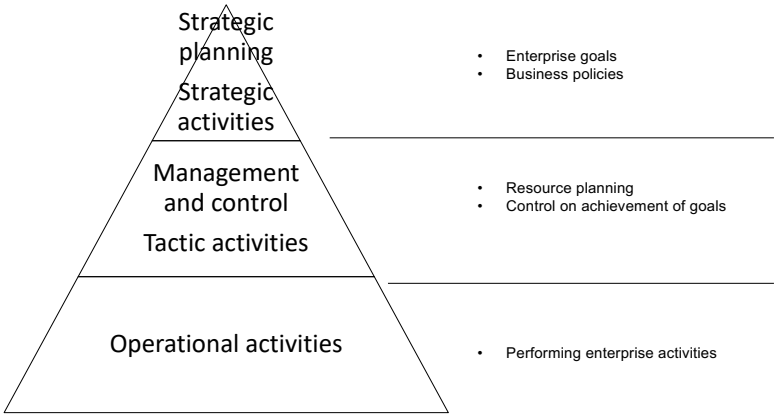


Figure 1.2: Information systems in organizations

- Send information.
- Present information.

These activities do not necessarily require IT tools. Information can be managed implicitly, i.e., based on the experience and skills of individuals, when it comes to activities that are difficult to replicate, or explicitly, but not supported by IT (manual management). Or again explicitly and supported by IT, that is, in a way that allows information to be organized and retrieved in an efficient and easily repeatable way.

In an organization, a **Process** can be defined as the set of activities that the organization as a whole carries out to manage the life cycle of a resource, or of a homogeneous group of resources, to achieve a defined and measurable result (product / service). Activities in processes are based on the use of information that is necessary for their execution and produce data that can be used within the same process or in other processes.

Processes can be classified according to different classification models, in the following we refer in particular to the well-known Anthony's pyramid.

The *Anthony's Pyramid* [2] provides a hierarchical classification of processes and applications. First of all, in Anthony's model, three levels are distinguished, linked to the structure of the organization (see Fig. 1.2: an *operational level*, in which the company's operational activities are considered, a *management and control level*, also known as directional control, in which tactical activities are considered such as the planning of available resources and control over the achievement of predefined objectives, and a *level of strategic planning*, where the activities are related to the choice of corporate objectives and the definition of corporate policies.

Examples of processes in a public administration, such as a municipal administration, are the following (broken down by level):

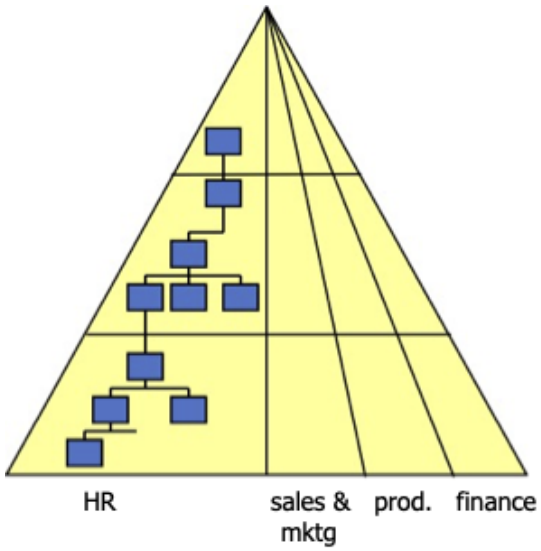


Figure 1.3: Anthony's pyramid

- *Operational*: accounting of citizens' payments, road maintenance interventions.
- *Control*: control of payments, reminders, monthly comparisons between expected and actual revenues, and pollution monitoring.
- *Strategic*: verification of costs and revenues relating to social services, definition of new tariffs, regulatory plans.

At a bank, examples of three-level processes are as follows:

- *Operational*: management of current account movements, ATM withdrawals (ATM - Automatic Teller Machine).
- *Control*: review of overdrafts, granting of a mortgage.
- *Strategic*: verification of the progress of a service, decision to open new services.

At a company, examples of processes are as follows:

- *Operational*: cost registration of orders, warehouse management.
- *Control*: control of weekly differences between budget and final balance.
- *Strategic*: choice of the most convenient market areas.

In Anthony's model (Fig. 1.3), the hierarchical model of the organization is the basis for representing processes, and for linking processes at different levels. Processes are partitioned considering the different sectors in an enterprise. Sectors can be generic, i.e., present in different enterprise domains of activity, such as Human Resource Management and infrastructure management (sometimes referred to as self-administration or support activities) or they can depend on the type of enterprise domain, such as, for instance, Manufacturing, Utilities, Banking and Insurance. On one hand, according to the model for each sector, all three levels are present. From top to bottom, the Strategic level drives the processes in the Control level, defining enterprise-wide strategic goals that are driving Control level activities for planning and control. Control level activities manage operational level activities, dealing with possible exceptions and controlling performance objectives wrt to plans. The interaction among levels is in terms of exchanging data, which allows the management of that sector of the enterprise. At each level, processes are organized hierarchically, with processes decomposed into more detailed processes. On the other hand, information flows also horizontally from one sector to another, sharing common information (e.g. the address of a customer) and triggering processes from one sector of activity in a company to another.

1.3 Data processing

The hierarchical representation of processes is related also to different information needs at each level. The different information characteristics can be classified according to the following dimensions:

- Time scope
- External / internal perspective
- Discretionality / Judgement
- Repetition
- Predictability
- Involved organizational roles
- Types of data

In general, in an information system, we can distinguish between two types of information processing: *operational systems* (often referred to also as OLTP - On Line Transaction Processing - systems) and *decisional systems* (often referred to as OLAP - On Line Analytical Processing - systems).

The purpose of the *operational systems* is to carry out basic, daily, normal administration operations, such as the management of transactions and office work, accounting, in general, the processing of what are called business situations. The operational information system corresponds to the operational level in Anthony's pyramid and to their management at the control level.

Decisional systems are information systems supporting corporate decision-making and strategic activities, in response to the need to exploit data assets to identify useful information for decision-making. As the amount of data available has grown, so has the demand for strategic information extraction. Information systems respond to this need with tools that have evolved from the original reporting tools or tools such as spreadsheets. Today, there are sophisticated functionalities of data extraction and analysis for planning and strategic purposes. With respect to this point, it is worth noting that in recent years there has been a strong push in the adoption of Artificial Intelligence (AI) techniques within information systems. This has made it possible to develop systems capable of making decisions supporting and sometimes replacing humans in decisions.

The term *transaction* in an information system refers to a very broad category of concepts which includes:

- Exchanges or contracts between the company and the external environment or between units of the same company with a specific objective (for example, receiving customer orders, sending orders to suppliers, etc.).
- Transformations, i.e. operational activities that contribute to the production of goods and / or the provision of services.
- Handling of physical objects within the company (for example, assembly of an electronic board) or between the company and the outside world (for example, shipping a piece from the warehouse to the customer).
- Certification of events, such as the insertion of a new product in the catalog or the payment of an invoice.
- At the database level, a set of elementary operations that modify the state of the database, leaving the data in a consistent state (for example, execution of a bank transfer, which includes various reads and writes on the databases which ultimately must terminate correctly with a 'done' or 'not done' result). This type of transactions in the context of the technologies of the database management systems (DBMS - DataBase Management System) is called ACID transactions (Atomic, Consistent, Isolated, Durable).

Transactions are often certified by the creation of a document, paper or electronic, which testifies that they have been carried out.

OLTP systems deal with operations characterized by a large number of short transactions and are focused on very fast query handling, maintaining data integrity in multi-access environments and ensuring system efficiency and effectiveness, for example, by measuring efficiency based on the number of transactions per second (throughput). Detailed and current data is stored in the OLTP data management systems. OLTP systems are suitable for managing processes at an operational and control level.

OLAP systems on the other hand, process historical (or archival) data. The OLAP paradigm is characterized by few complex transactions and queries that

	OLTP	**OLAP**
user	employee (many)	manager (few)
function	daily operations	decision support
design	application-oriented	object-oriented
data	current, up-to-date, detailed, homogeneous	historical, aggregated, multidimensional, heterogeneous
use	repetitive	ad-hoc
access	read / write	read
unit of work	transaction	complex query
metric	throughput	response time

Table 1.1: OLTP vs OLAP

require data aggregation. OLAP systems have response time as a measure of efficiency as they usually need to process a large amount of data. OLAP applications are often based on Data Warehousing techniques. An OLAP system stores data in aggregate format, stores historical data, archived according to multi-dimensional schemes (generally organized according to the analysis dimensions). Queries often access large amounts of data to answer complex questions such as, for example, "What was the company's net profit in a certain geographic area in the past year?". OLAP systems are used for decision support-oriented data processing, so they are adequate for functionality located at the management and strategic level of Anthony's pyramid. In fact, they allow to carry out complex and ad-hoc operations, in which each single operation can involve a lot of aggregate, historical, and generally even non-current data, and for which the ACID properties are not relevant because the operations are read-only.

The different characteristics of OLTP and OLAP systems are summarized in Tab. 1.1.

1.4 Data-driven management

In all processes described above, data are continuously collected. Digital transformation is making processes more and more supported by digital technologies and therefore the volume of collected data is increasing.

These data are valuable for the management of a company as they can be exploited to gain insight into the activities of a company. The term *Data-Driven Decision Making*, or DDDM, is used to define the process of making decisions based on data. Data is the basis of decisions in many companies today and is often considered one of the main factors responsible for their success.

As shown in Fig. 1.4, raw data can be transformed into valuable information for managers. Data can be processed and analyzed to better understand the activities of the company. For instance, they can be used to identify which are

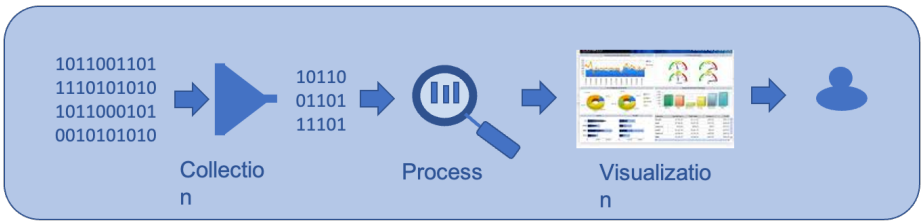


Figure 1.4: Data-driven management. Source https:www.smartsheet.com/data-driven-decision-making-management

the most profitable activities, to analyze processes and improve their efficiency, to understand the preferences of clients, and to recommend products to them. In fact, data can be used to provide new insights about the activities of a company. To provide insights to managers it is also important to be able to present them in a synthetic and understandable way. These analyses are based on the availability of a large amount of data, collected both internally in the company and from external sources. Therefore data are an essential asset in a company as they can provide valuable information for reaching, interacting with and retaining customers and for performing their activities efficiently.

Data-Driven Decision Making, or DDDM, is the process of making decisions, based on data. Decisions are data-driven in many companies today and this is often considered one of the main factors responsible for their success.

On the other hand, to achieve such results, it is important that the data on which decisions are made are properly collected and managed.

The use of Data-Driven Decision Making by sector and by country is presented in Fig. 1.5 and Fig. 1.6. In Fig. 1.5, it is clear that DDDM is frequently adopted in organizations providing services and utilities, as it allows analyzing systematically customer trends and needs, but it also becoming broadly adopted in other sectors, also due to the availability of technologies to monitor the environment and machines with sensors.

In addition, companies are increasingly assessing the value of their data. In Fig. 1.7 the evaluation of data monetization and data quantification in organizations worldwide in 2020, by sector. According to 60 percent of respondents, the banking sector is most likely to monetize data assets and insights through its products and services in 2020. In order to monetize data effectively, global organizations should have a mature approach to designing products and services that capture new data. This potential of monetization of data, however, is often not yet reflected as a value in accounting systems.

1.5 Data governance

As mentioned above, data are an essential asset for companies both to execute activities and to support management control and decision.

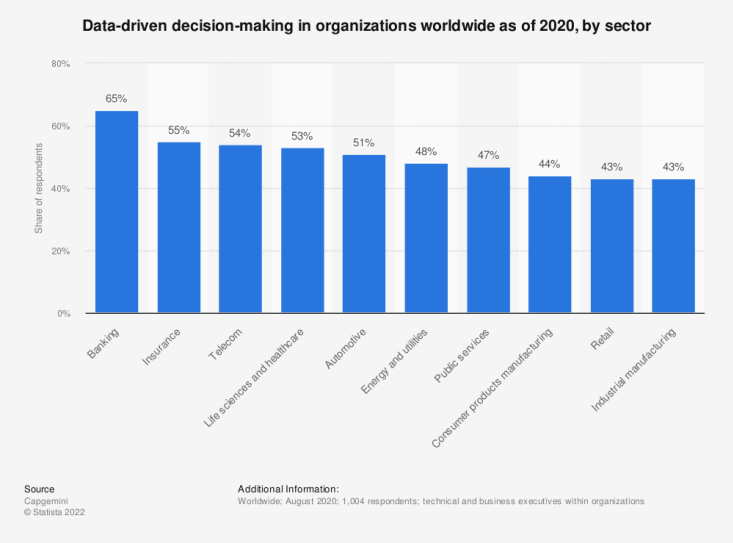


Figure 1.5: Data driven decision making in different sectors

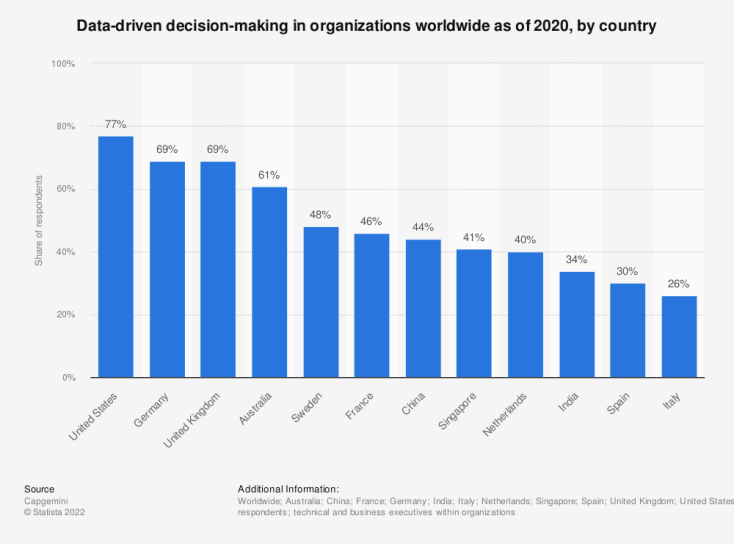


Figure 1.6: Data driven decision making in different countries

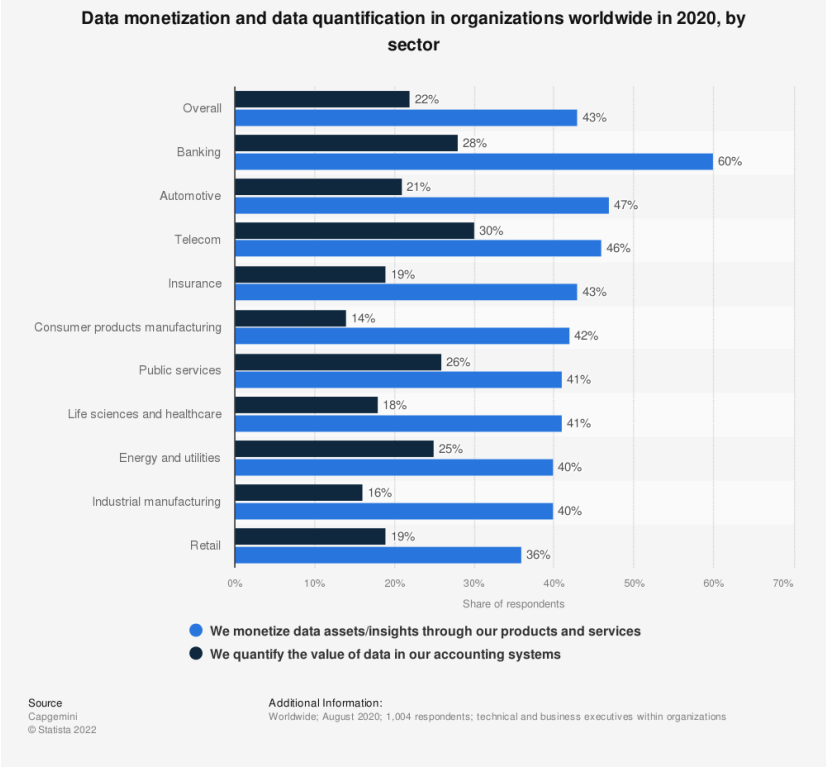


Figure 1.7: Data driven monetization and data quantification

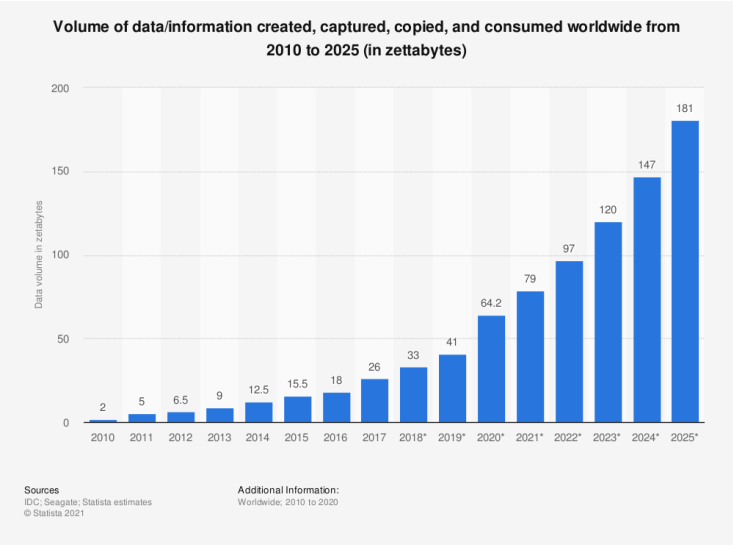


Figure 1.8: The growth of big data since 2010 (source Statista)

Many digital transformation projects focus on digitalizing processes in order to be able to manage them in an effective way. Digitalized activities produce data in large quantities. Such data include the actual data being used in the process and data logs generated by logging activities which are essential to monitor the performance of the system and also accesses to the system. Data used in processes can include many different types of information, such as records of data, textual information, media such as audio, images, videos, and numeric information. Such information can be originated internally by the company during its activities or can be acquired from external sources, such as data providers, social media, external sensors, and so on. The amount of data being produced is increasing exponentially over the years and it is posing significant challenges (see Fig. 1.8 and Fig. 1.9[1])

As an asset for an organization, data have some specific characteristics, which have been studied by Moody in his research work [16], who formulated some general qualitative rules about the value of data. We discuss here the most relevant ones for data governance:

- *The Value of Information Increases With Use*: while other resources are consumed when they are used and cannot produce further value, data instead is getting value the more it is used (see Fig. 1.10). Most of the costs related to data are related to their collection, storage and maintenance, while the costs for using it are very limited. In general, information has no value on its own, it is valuable only when it is actually been used.

[1] A zettabyte ZB is defined as $(1000)^7 bytes$

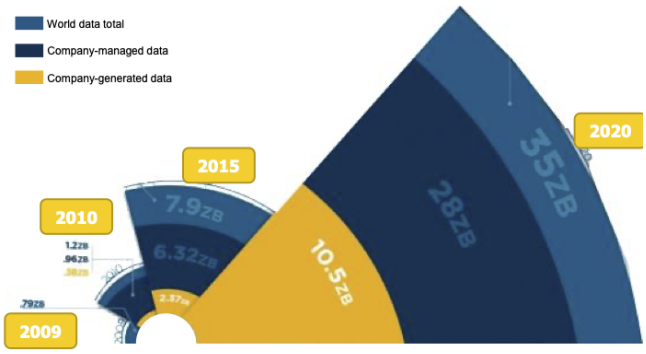


Figure 1.9: The growth of big data in organizations (source CSC)

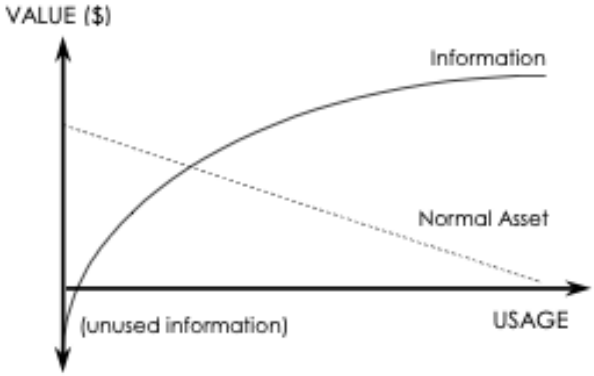


Figure 1.10: The value of information increases with use [16]

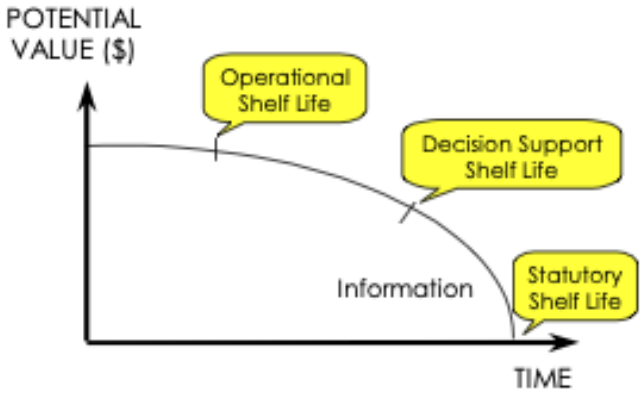


Figure 1.11: Information is perishable [16]

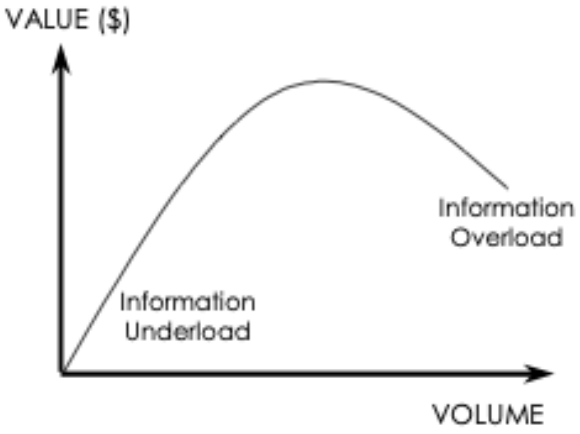


Figure 1.12: Value of data vs quantity [16]

- *Information is perishable*: Information is first used in the operational system, then for decision support, and eventually its value decreases over time to become useless (Fig. 1.11).
- *More is not necessarily better*: larger quantities of data are not necessarily producing more value for a company, as illustrated in Fig. 1.12.

  Some of the collected data may be never used, or they become useless over time. Already in 2002, studies mentioned that 69% of the data stored by companies can be called "data debris" as they have no current business or legal value.

In this paragraph, we focus on two aspects that are critical for being able to make data a real resource for companies: the ability of the company to know its data and the need to follow all needed legal and internal regulations which can

mandate to follow given procedures requiring the preservation of data over time, computing indicators, in addition to defining security and privacy measures.

*D*ata governance has the goal of keeping data under control and secure.

As defined in [19], it has the goal of overseeing the integrity and security of data owned and used by an organization, as well as managing that data's availability and type of use.

So, in fact, there are two main aspects: to be able not only to store data, but also to have information about the data itself. In many cases, we will refer to this type of data as *metadata*, i.e., data about data. In addition, data must be managed as any other resource in the company, and it has to follow its life cycle from acquisition and use, until disposal. It has also to be curated, so that it can be used appropriately in an organization. In addition, it is necessary to comply with laws governing security and privacy and other internal and external regulations that enforce storing and preserving data about the company activities, to provide them to authorities for audits, legal needs, and statistical purposes. All these aspects result in a set of procedures, rules, and processes that enable the storage and use of data in an appropriate way.

Several aspects need to be considered in data governance, and they enable the organization to answer some essential questions about their data [19]:

- *Is the data protected?* laws and regulations enforce data protection, and in addition, organizations need to protect data against possible attacks. It must noted that laws vary in different countries and this is posing additional issues to companies operating in international contexts.

- *How long should data be retained?* Also in this case there may be laws requiring the preservation of data, as well as internal regulations in the company.

- *Where is it located?* locating data presents two main issues: on one hand, the organization infrastructure must be able to find relevant data when they are needed and be aware of their existence to exploit them in decision-making. On the other hand, there is a need to manage also the physical location of data, as it might be subject to different laws in different countries.

- *Who has access to data?* It is essential to define who are the users who can access the data, the rules for accessing it, and the privileges assigned to each user in processing data and managing them. The policies for privilege assignments must be defined.

- *What does it contain?* To be able to use data it is necessary to be able to understand it. This aspect concerns not only the single values of data but also entire data collections, which must be properly described to be correctly used in operational activities, and in analysis and Artificial Intelligence (AI) applications and Machine Learning (ML).

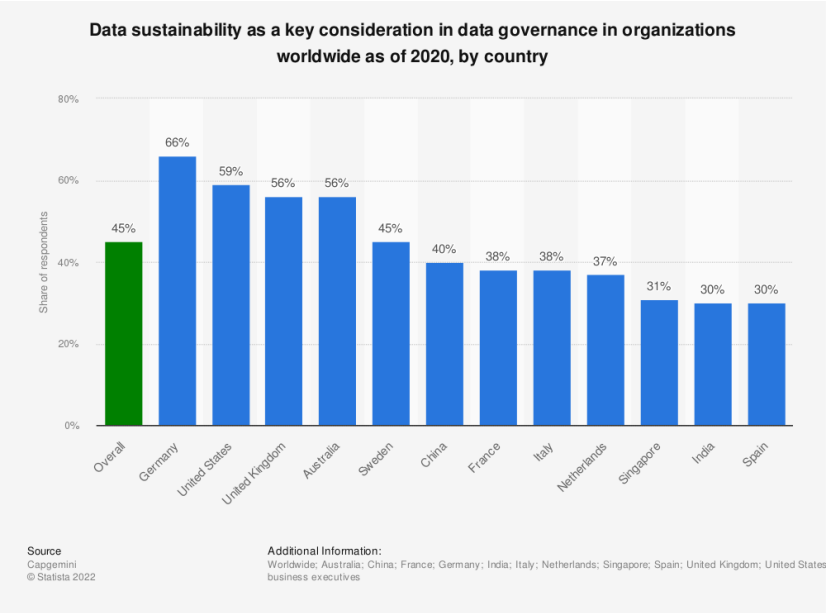


Figure 1.13: Data sustainability in different countries (source Statista)

- *How can the data be used to produce a competitive advantage?* In addition to storing data, processes, and services for using them must be defined to extract value from data.

Another important issue is the concern about *data sustainability*, namely, the ability to ensure data production, storage, and use are sustainable. As data governance processes are adopted, and the number of laws and regulations increase, it is also important to be able to adopt technologies to facilitate data governance and prioritize according to needs. Data sustainability is being increasingly considered in different countries, as shown in Fig. 1.13.

Different aspects related to data governance will be analyzed in this text, both as enablers and considering sustainability over time:

- Data structures, metadata, storage, and interchange formats.
- Data quality.
- Architecting data services: Datawarehouses, pipelines, datalakes, data exchange.
- Security and privacy, ethical issues.

1.6 Scenarios

Three main application scenarios will be considered: business administrative data, extraction of information from social media, and smart and mobile objects.

In this section, a short introduction is provided to the three application scenarios discussed in the rest of the text.

Administrative data (Fig. 1.14).

Supplier_Number	Supplier_Name	Supplier_Street	Supplier_City	Supplier_State	Supplier_Zip
8259	CBM Inc.	74 5th Avenue	Dayton	OH	45220
8261	B. R. Molds	1277 Gandolly Street	Cleveland	OH	49345
8263	Jackson Composites	8233 Micklin Street	Lexington	KY	56723
8444	Bryant Corporation	4315 Mill Drive	Rochester	NY	11344

Figure 1.14: Example of administrative data (source [15]

Data in social media (Fig. 1.15)

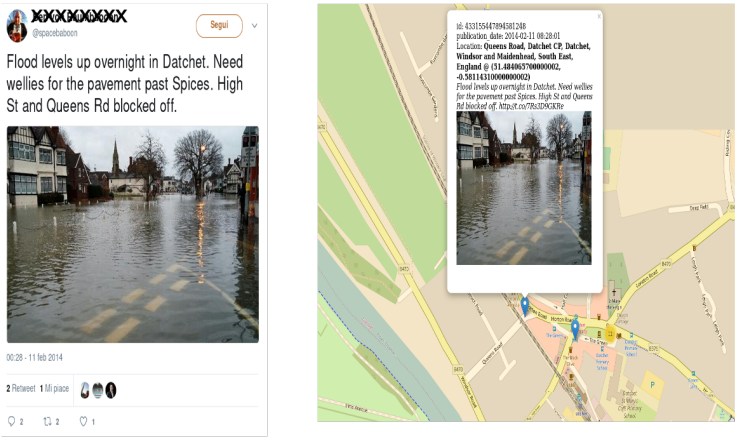


Figure 1.15: Example from Twitter and E2mC project

Data in smart and mobile objects (Fig. 1.16)

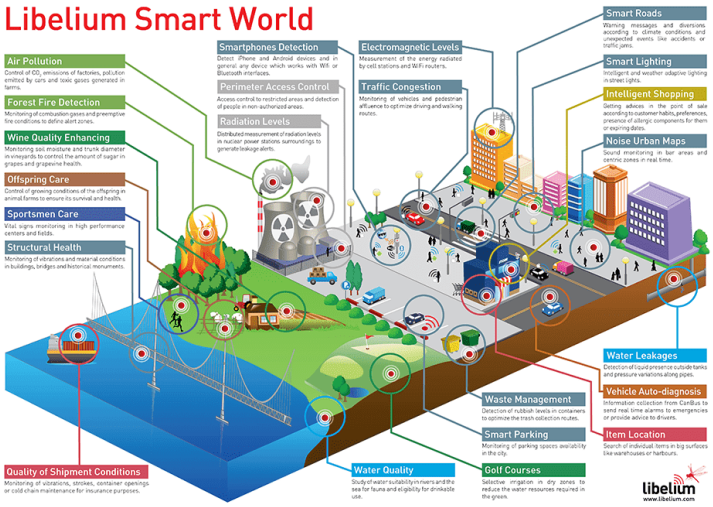


Figure 1.16: Example of smart world from Libelium

1.7 Suggested exercises

- Search for the other Moody's Laws
- Search on Statista (or Google): Distribution of the expenditure on big data in your area of interest

Chapter 2

Data Management

Barbara Pernici, Carlo Batini[1]

2.1 Introduction to data

2.1.1 Structured and unstructured data

Structured data Structured data have a regular data structure common to all available information of the same type, composed of structured elements.

Unstructured data Unstructured data include free text, irregular structures, other types of data (images).

Metadata **Definition**: *Metadata* = data on data. Metadata describe the data, their structure, and possible values. They are data stored in the system.

Storing data Data are sequences of bits. To make them manageable and understandable we need to give them some form of structure. In this chapter, we focus on structured data, while unstructured data will be studied in the following chapters. The relationship between bits, records, and files is illustrated in Fig. 2.1.

The following terms are introduced:

- *Database*: Group of related files
- *File*: Group of records of same type
- *Record*: Group of related fields
- *Field*: Group of characters as word(s) or number(s)

[1]Part of this chapter is based on materials of the Course on Database Design (C. Batini)

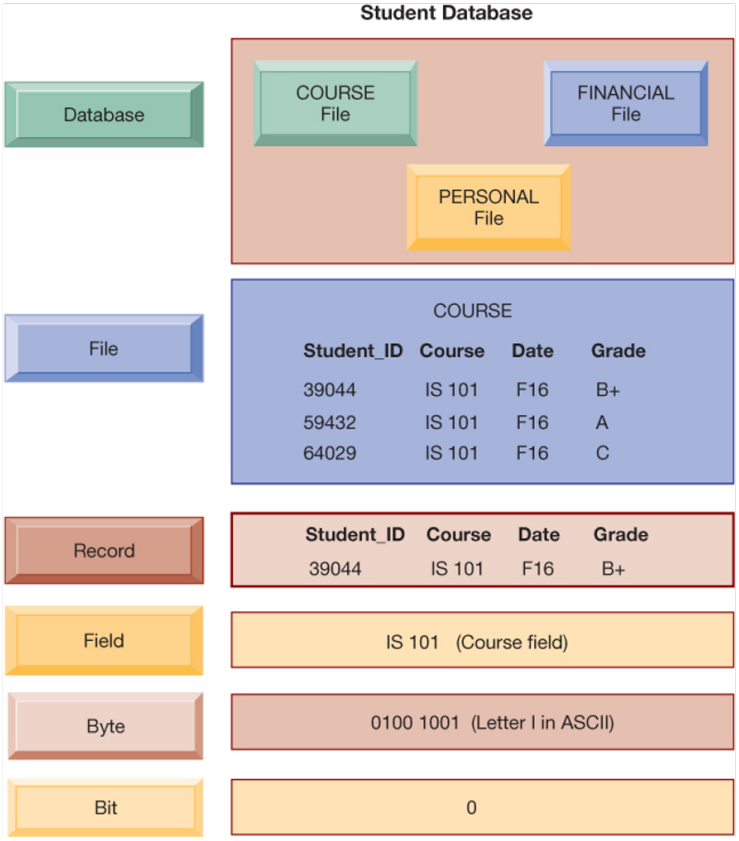


Figure 2.1: Data in an information system (source: [15])

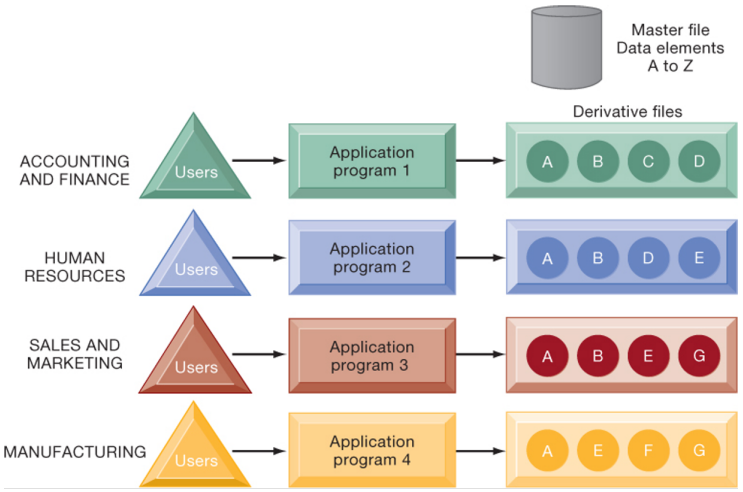


Figure 2.2: Storing data in files for different applications (source: [15])

As shown in the figure, in the database, we store information on different elements, and in files we group records of the same type. We will use the following terms to describe the elements:

- *Entity*: Person, place, thing on which we store information (e.g., Course, Financial, Personal in the figure)
- *Attribute*: Each characteristic, or quality, describing an entity (e.g., StudentId, Course, Date, Grade in the example)

As seen in the introduction, data may be used by different users, who are performing different operations on them (e.g., several business functions may share customers' data). If we consider them separately, we may access and manage data separately, as shown in Fig. 2.2.

Problems with the traditional file environment for storing data separately arise:

- Files maintained separately by different departments
- Data redundancy
- Data inconsistency
- Program-data dependence
- Lack of flexibility
- Poor security
- Lack of data sharing and availability

To avoid these problems the DataBase Management Systems (DBMS) technology has been developed, starting in the '70ies.

A *Database* serves many applications by centralizing data and controlling redundant data.

Definition - A database is a collection of data:

- used to represent information of interest to different sectors of an organization,
- shared between different software applications and different users of the organization,
- where each information of interest is represented only once in the collection of data.

A *Data Base Management System (DBMS)* is a software system able to manage requests of users and software applications in terms of queries and transactions, to access databases, namely collections of data that are:

- *Large* (bigger, often much bigger than the main memory available).
- *Shared* (used by various software applications and users).
- *Persistent* (with a lifespan not limited to single executions of the applications that use them).

A Database management system (DBMS) interfaces between applications and physical data files, separates logical and physical views of data, solves problems of traditional file environment, controls redundancy, eliminates inconsistency, uncouples programs and data and enables the organization to centrally manage data and data security.

An example of multiple views on the same data is provided in Fig. 2.3.

A basic architecture using a DBMS is a *client-server architecture* (Fig. 2.4) allowing sharing data for different clients.

As, in general, several applications may use the same database and present the data with different interfaces to their clients, and application layer can be introduced, as shown in Fig. 2.5, which distinguishes between three logical layers (also called *PAD* architecture):

- *Presentation*: the presentation layer is devoted to present to users the data and related functionalities (to retrieve data, to insert new values, and so on) using different clients, i.e., interfaces (e.g., an input form, a table, a graphical presentation of the data).
- *Application*: it provides the logic of the operations that can be performed on the system, according to business rules (e.g., how to manage the insertion of a grade for a student).
- *Data*: this layer contains all functionalities needed for managing data (Creating, Reading, Updating, and Deleting data, also known as *CRUD operations*), giving access to authorized users, and storing data safely.

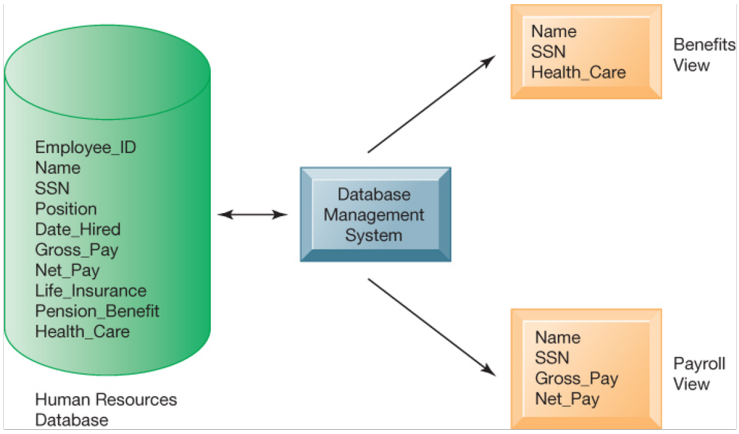


Figure 2.3: Multiple views on data (Source: [15])

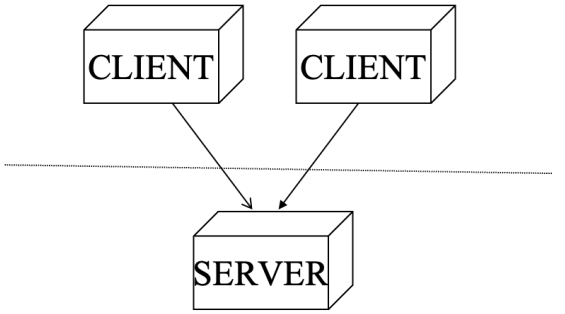


Figure 2.4: Client server architecture

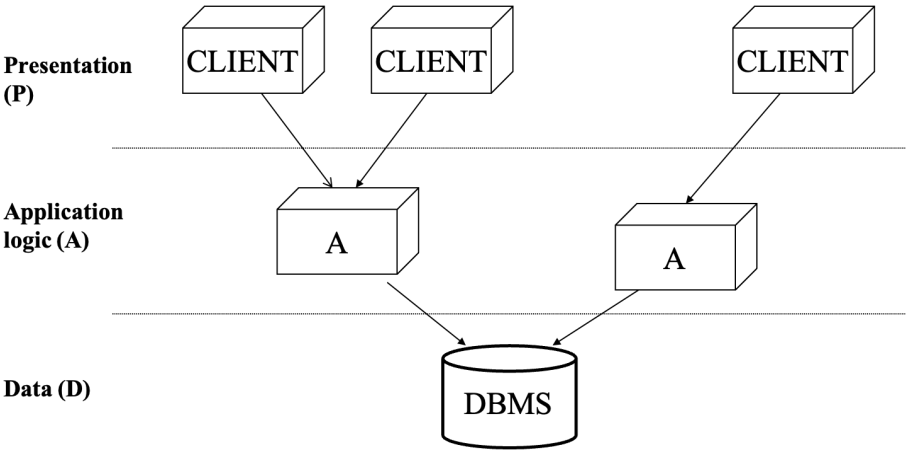


Figure 2.5: Three layered architecture

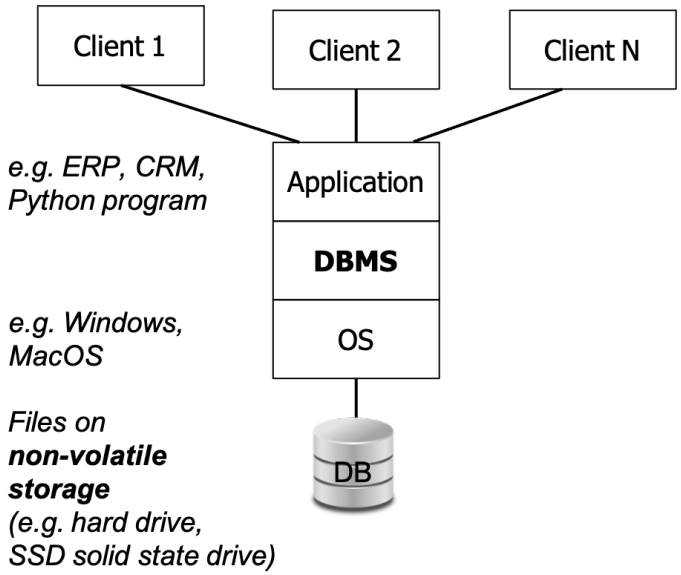


Figure 2.6: System architecture

2.2 Database Management Systems (DBMS)

The DataBase Management Systems technology (DBMS) is a technology that allows sharing data among different applications.

Functionalities and components of a DBMS are the following:

- Querying and reporting
- Data manipulation language
- Data definition capability
- Data dictionary
- Manage access by several users
- Guarantee persistence
- ...

A basic intro to DB technology is provided in Fig. 2.6.

Databases are based on *data models*. Data models can be *conceptual models* (e.g., the Entity-Relationship model), defining conceptually the contents of the database, and *logical models*, giving the database its actual structure. Conceptual data models are used for designing databases, while logical models are used in DBMS as a basis for managing data.

In this chapter, we study the *relational model*, described in the next section, which is the model which is usually adopted in DBMS for structured data, and it is based on the concept of *relational table*.

For each database, it is necessary to define a *schema*, which is the structure given to data in the database.

DBMS provide a *data definition language* to define the structure of its data and a *query language* (or better manipulation language), that allows performing CRUD operations on the database.

Requests of users and software applications are of two types:

- *Queries*, that retrieve from the database data that match certain selection criteria expressed in the query. As such, queries do not change the state of the database.
- *Transactions*, that insert, delete, and update data in the database. As such, transactions change the state of the database.

2.3 The relational model

2.3.1 Tables

The relational model is based on a table structure for data.

Definition - The *schema of a relation R* (or relation schema) is the set of concepts represented in the relation; it is made of the *relation name R* and its properties, also called *attributes*. For the example of relational tables illustrated in Fig. 2.7, we can define two relation schemas, corresponding to:

- Suppliers, with the relation SUPPLIER, with attributes Supplier_Number, Supplier_Name,
- Parts, with the relation PART, with attributes Part_Number, Part_Name, Unit_Price, Supplier_Number.

Notice that we can represent the schema in a synthetic form, e.g., the relation schema associated with the Supplier as
SUPPLIER(Supplier_Number, Supplier_Name, Supplier_Street, Supplier_city, Supplier_State, Supplier_Zip)

Definition - The *schema of the database* (or database schema) is the set of relation schemas of the database.

In the example, we have a unique database schema made of:

SUPPLIER(Supplier_Number, Supplier_Name, Supplier_Street, Supplier_city, Supplier_State, Supplier_Zip)
PART(Part_Number, Part_Name, Unit_Price, Supplier_Number)

Notice that the values represented in relations are not part of the schema.

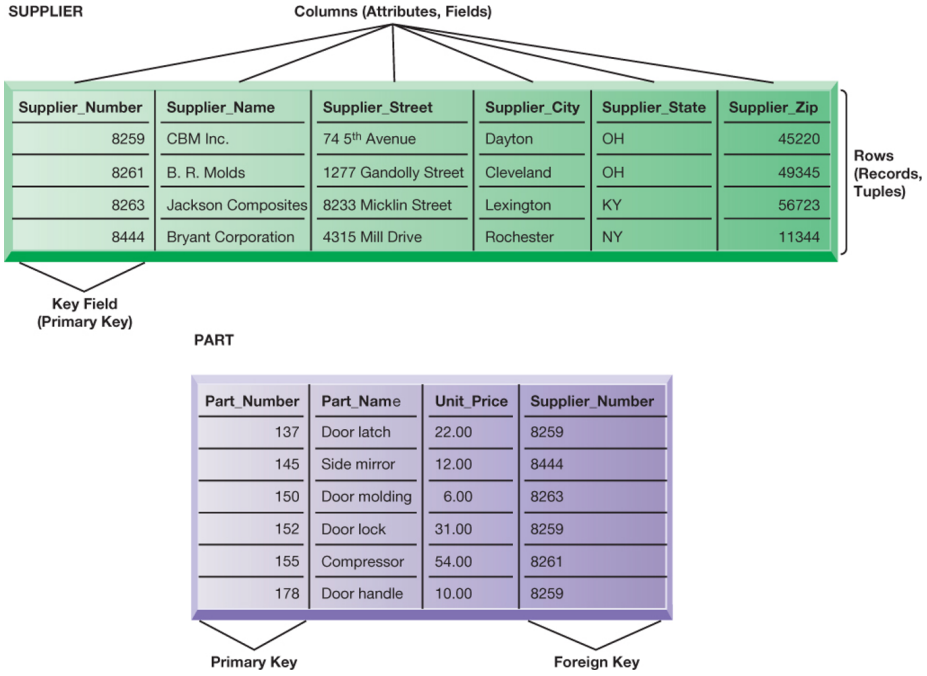


Figure 2.7: Example of relational tables (source: [15])

Definition - Given a relation schema R, an *attribute* is defined as a pair <Attribute Name, Domain> where Attribute Name is the name of the attribute and domain is the set of values that the attribute may have in the relation instance.

For instance, the domain of Supplier_Number is an INTEGER.

Definition - Given a relation schema R(A1, A2, . . . , An), a *tuple* is defined as <A1: v1; A2: v2; . . . ; An, vn> where, for each Ai, vi is a value in the domain of Ai.

A tuple of the PART relation is
<137, Door latch, 22.00, 8259>

Summary of relational database terminology

Database schema – When the database is made of several tables, the database schema corresponds to the set of their relation schemas.

Relation instance – The relation instance represents the values of concepts in the relation (e.g., the Part Name is Latch, the Unit Price is 22.00); values can change over time (e.g., a new part could be added).

Database instance – It is a set of relation instances all related to the same database.

Tuple – It is a specific set of values in the relation instance corresponding to a single row in the table.

2.3.2 Keys

Definition - A *tuple constraint* over a relation R expresses a property that must be true for all tuples of R.

For instance, if we consider a relation Exams(StudentID, CourseID, Grade, Laudem), a tuple constraint states that the Laudem value can be given (value 'L') only if (grade =30), i.e., if (grade<30) then Laudem <> 'L'.

Key constraints

Key constraints are one of the most important, perhaps the most important concept, in the relational model. They play in the relational model the role of the identifier for tuples.

Definition - Given a relation schema R defined on attributes A1, A2,..., An, said B and C two disjoint subsets of the attributes, we say that a *functional dependency* holds between B and C, and write

B → C

when for each instance of R, to each set of values <b1, b2, .., bk> of attributes in B, a unique set of values <c1, c2, ..., ch> of attributes in C corresponds.

For instance, in the relation schema PART(Part_Number, Part_Name, Unit_Price, Supplier_Name), for all of its relation instances, the following functional dependency hold

Part_Number → Part_Name, Unit_Price, Supplier_Name

(we assume here that a part with a given number has only one supplier).

Definition - Given a relation schema R(A1, A2, ..., An), when an attribute Ai or in general a group of attributes Ai1, ...,Aik of R is such that

1. Ai1, ...,Aik → all other attributes in R

and

2. no subset of Ai1, ...,Aik has the same property

we say that Ai1, ...,Aik is a *key* of R.

The concept of key is the most important concept in the relational model. The value of a key allows to *uniquely identify objects* of the real world represented in R.

For instance, in the PART relation above, Part_Name is a key for the relation.

Definition - We call *primary key* a key whose values are always specified, namely, are different from null.

We want to emphasize a point that we addressed before in the lesson. Look at the following relation (Fig. 2.8).

Student

StudentId	Social Security Number	Surname	City of Birth	Country of Birth
13	BFG3RTK	Batini	Rome	Italy
21	null	Xu	Harbin	China
32	GRT5YHF	Smith	Paris	France
48	SDE5IKL	Wang	Harbin	China

Figure 2.8: A STUDENT relation

In the specific relation instance shown above also the following functional dependency holds

Surname → Student Id, Social Security Number, City of Birth, Country of Birth

but this is not true in general, for all possible instances!!!

Also Social Security Number cannot be a primary key since it can have null values.

We may represent the primary keys of the two relation schemas underlying all the attributes in a relation schema that take part in the key. For instance:

SUPPLIER(Supplier_Number, Supplier_Name, Supplier_Street, Supplier_city, Supplier_State, Supplier_Zip)
PART(Part_Number, Part_Name, Unit_Price, Supplier_Number)

Notice now the following rule in the referential integrity constraint

PART(Supplier_Number) → SUPPLIER(Supplier_Number)

The supplier Number in the PART relation *must* exist in the supplier relation. We call this a *referential constraint* and Supplier_Number is a *Foreign Key* in the part relation, i.e., given a value of Supplier_Number in PART, there must exist a tuple in the SUPPLIER relation with that number as a Primary Key.

Keys play a very important role in relational models: they avoid the insertion of duplicate tuples (no two tuples with the same Primary Key can be inserted in a relation) and Foreign Keys guarantee the consistency of the database, as they guarantee that a tuple with that value is present in another linked relation.

2.4 Structured Query Language (SQL)

Three basic operations are defined to query data[2]:

- SELECT Creates a subset of data of all records that meet stated criteria.

[2]https://www.w3schools.com/sql/

- PROJECT Creates a subset of columns in a table, creating tables with only the information specified.
- JOIN Combines relational tables to provide users with more information than is available in individual tables.

SQL query structure

In SQL (Structured Query Language), queries are expressed as:

```
SELECT  List of attributes
FROM List of tables
WHERE condition
```

The SELECT corresponds to the PROJECT operation illustrated above, as it specifies which attributes must be in the result.

FROM and WHERE clauses express the conditions for selecting elements from the tables. Some conditions allow specifying JOINS; as described below.

The result of a SQL query is always a relational table, with a schema as specified in the SELECT clause.

Examples

```
SELECT  Supplier_Name, Supplier_City
FROM SUPPLIER
WHERE Supplier_State = 'OH'

SELECT  Supplier_Name, Supplier_City
FROM SUPPLIER
WHERE
Supplier_State = 'OH' AND
Supplier_Number > 8260
```

In the two queries, Suppliers' Names and Suppliers' City define the structure of the resulting table. The data are selected from the Supplier table, and a selection criteria is defined. The selection criteria are logical conditions on the values of the attributes (e.g. Supplier_State='OH'). Several conditions can be combined with AND and OR logical operators, if necessary with parentheses (the usual precedence rules of logical operators are valid, i.e., AND clauses have higher priority than OR clauses if no parenthesis is used). The results of the selections are as follows, respectively:

when you need to retrieve information from more than a table you need to join them. For example, in Fig. 2.9, we want to retrieve information about the suppliers of the parts with id 137 or 150. In this case, we want in the result the Part_Number, Part_Name, Supplier_Number, Supplier_Name

To solve this query we need to exploit the Supplier_Number, which is provided in both tables.

Supplier_Name	Supplier_City
CBM Inc.	Dayton
B. R. Molds	Cleveland

Supplier_Name	Supplier_City
B. R. Molds	Cleveland

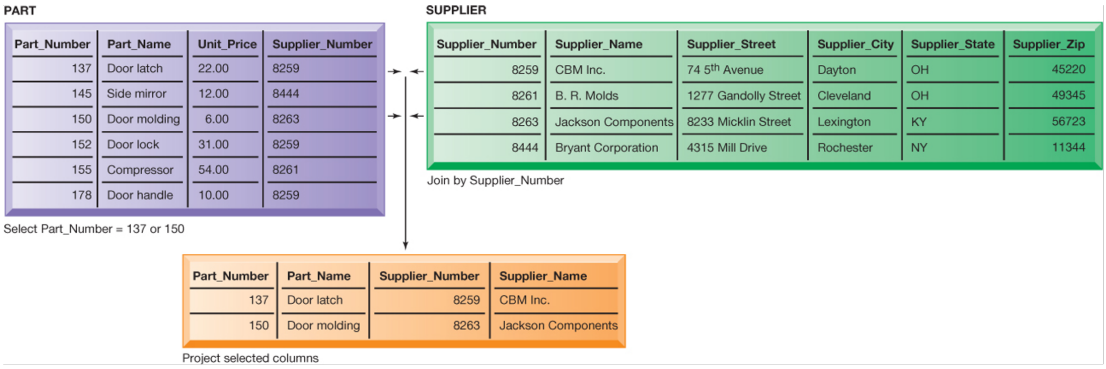


Figure 2.9: Joining tables

```
SELECT  P.Part_Number, P.Part_Name, S.Supplier_Number, S.Supplier_Name
FROM SUPPLIER S, PART P
WHERE
S.Supplier_Number = P.Supplier_Number
AND
(P.Part_Number = 137 OR P.Part_Number = 150)
```

2.5 An introduction to SQL - Create Table

```
CREATE TABLE Persons (
    PersonID,
    LastName,
    FirstName,
    Address,
    City
);
```

2.6 CRUD operations

```
INSERT INTO Supplier (Supplier_Number, Supplier_Name)
VALUES ('9445', 'ACME S.p.A');
```

with all VALUES or a subset
Hint: SELECT * From Supplier; prints all inserted tuples in a table
DELETE, UPDATE

2.7 Adding constraints

PRIMARY KEY
FOREIGN KEY
NOT NULL
CHECK AGE>18

```
CREATE TABLE Persons (
PersonID PRIMARY KEY CHECK (PersonID > 100),
LastName,
FirstName,
Address,
City
);
```

How to modify a table, including also constraints
ALTER TABLE
ALTER TABLE PERSONS ADD Age
ALTER TABLE PERSONS ADD Age CHECK (Age >17);

2.8 Suggested exercises

- Python Notebook for this chapter:
  `https://colab.research.google.com/drive/1T9BI5aeNj1nh9ycUiFMHkgjEQO4sZUep?usp=sharing`
- The notebook above contains at the end the following challenge:

```
- Create 3 Tables Customer, Product, Sale.
Each has a key, and other attributes for describing the items
(e.g. Name, Age for customers, Brand for Products)

Sales contains tuples indicating sales of products to customers,
with Quantity and Date
- Insert 2 customers and 3 products
```

```
- Insert a tuple into Sale
- Display the age of Customers and Date for each sale
```

2.9 Resources

- SQL interactive exercises: `https://www.w3schools.com/sql`
- SQLite web site: `https://www.sqlite.org/index.html`

Chapter 3

Data quality

Cinzia Cappiello, Barbara Pernici

3.1 Introduction

Nowadays, organizations often adopt a data-driven management that is characterized by the practice of collecting data, analyzing them, and basing decisions on insights derived from the information [1] (see Ch. 1). It means that all the decisions the organization takes are based on some data analysis results. In order to make effective decisions, the analysis results should be as accurate as possible. Inaccurate results might be caused by inaccurate input data. This is called the "*Garbage-In-Garbage-Out (GIGO)*" phenomenon: errors in the input data might propagate in the data processing and negatively affect the output values. It is worth noticing that bad results can be also obtained due to errors in the method used to analyze the data. Therefore, as depicted in Figure 3.1, it is possible to claim that the success of data-driven decision-making depends on

- the quality of data collected
- the methods used to analyze data.

However, in this course, we do not consider the latter aspect and we focus only on the issues that might affect the input data. It is important to consider big but also small errors. Small errors are, in fact, sometimes ignored but they can cause the so-called "snowball effect": small errors propagate and have a big impact on the quality of the results. For example, a simple inattention has caused a loss of $125 million to NASA[2]. In September 1999, the Mars Climate Orbiter, a key part of the NASA program for the Mars discovery, vanished because it entered the Martian atmosphere well below its intended altitude. This was caused by a missing translation of the spacecraft's trajectory measure

[1] https://www.smartsheet.com/data-driven-decision-making-management
[2] https://sma.nasa.gov/news/safety-messages/safety-message-item/lost-in-translation

unit. This negligence seems to be a small step in the whole design of the probe, but instead, it caused the complete failure of the space mission. Similar Data Quality "horror stories" are frequent and they can be easily found on the Web.

To guarantee a certain level of quality in the data analysis process, we need to add a data preparation phase in which data must be transformed and cleaned. The data preparation phase is crucial and mandatory since real-world data are often incomplete, inconsistent, and contain many errors. Moreover, it is also a time-consuming task: data preparation, cleaning, and transformation comprise the majority of the work in a data mining application (80-90%) [4].

Quality of data is important also in Machine Learning. Machine Learning is based on training activities performed based on sample data that result in the creation of a learned model. The quality of the input influences the quality of the output also in this case. As many decision support systems are based on predictions performed with ML models, also in this case the effort in data preparation is significant, as in the previous case.

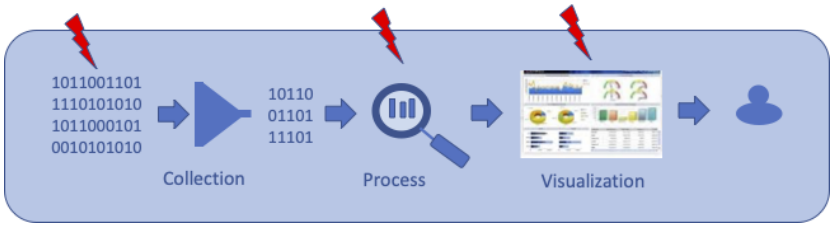


Figure 3.1: Data quality in DDDM

3.2 Data quality definition

Data Quality has been defined in numerous ways in the literature. The most common definition is "Fitness for use", which is the ability of a data collection to meet user requirements [20]. From this definition, it is possible to understand that the data quality evaluation depends on the processes/users that access and use data. Therefore, a source can be considered acceptable for one analysis and inappropriate for another analysis. From an information system point of view, it is possible to state that the databases of an organization are characterized by high quality if there are no inconsistencies between the values in the databases and their counterparts in the real world. Organizational data are a representation of real-world objects and events and they have to be aligned with them. An incorrect representation leads to poor-quality data.

3.3 Data quality dimensions

Data Quality (DQ) is not a synonym for data correctness, but it considers different aspects. The main issues to consider in data preparation activities are,

for example, related to missing values, duplicate data, inconsistent data, and outliers. To define data quality in a specific scenario, a *DQ model* has to be defined. A DQ model is composed of *DQ dimensions* that represent the different aspects to consider. Each DQ dimension can be associated with more than one *metric* that clarifies how the dimension can be assessed.

In the literature, a large number of data quality dimensions have been defined. A survey described in [20] identified 179 data quality dimensions. Such a long list has then been analyzed and eliminating redundancies and inconsistencies, a group of 15 dimensions has been selected. However, the dimensions that are often used in all the analyzed contexts are four: accuracy, completeness, consistency, and timeliness [4].

Accuracy Accuracy is defined as the closeness between a data value v and a data value v', considered as the correct representation of the real-life phenomenon that the data value v aims to represent. [4] For example, let us assume that v is equal to "Jhn" and v' is equal to "John": it is possible to claim that v is not accurate. In the literature, two types of accuracy are defined Syntactic accuracy and Semantic accuracy. *Syntactic accuracy* is the closeness of a value v to the elements of the corresponding definition domain D [4]. If v is any one of the values in D, then v is accurate otherwise it is not accurate. "Jhn" is an example of a lack of syntactic accuracy: "Jhn" is not included in the domain of names. *Semantic accuracy* is defined as the distance between a data value v and a data value v'. In this case, v is a value of the domain D. Semantic inaccurate values are those that despite belonging to the domain are not the correct ones. For example, if in a database it is written that the director of the movie "Dead Poet Society" is Spielberg, we have a semantic inaccuracy. In fact, Spielberg is a director and therefore the value belongs to the right domain but it is not the correct one. For the assessment of accuracy, it is possible to use, for example, a Boolean approach: we assign 0 to all the inaccurate values and 1 to the accurate 1. If we consider a table with N values, the accuracy of the table can be assessed as the ratio between the number of correct values and N.

Completeness The completeness of a table characterizes the extent to which the table represents the corresponding real-world [4]. It is related to the presence of null values and a simple way to assess completeness in a table is to calculate the ratio between the number of non-null values and the number of cells of the table. The problem in the assessment of completeness is that *sometimes the meaning of the null value is not clear.* The null values to consider for completeness are the ones related to values that exist in the real world but they are missing in the analyzed data set. It could happen that null values are instead caused by the fact that the value does not exist in the real world. For example, the telephone number of a person is null because the person does not own a telephone. In this case, the null value should be considered as a complete value since it represents the fact that the value does not exist in reality. In the design process of a database, these two situations should be distinguished. In

the latter situation, the values should not be null but a code able to represent the unavailability of the values should be defined (e.g., 'NA'). In summary, before assessing the completeness, it is necessary to understand which of the null values should be considered.

Consistency The consistency dimension captures the violation of semantic rules defined over (a set of) data items, where items can be tuples of relational tables or records in a file [4]. Semantic rules can be integrity constraints (e.g., functional dependencies) or business rules. Once the rules are defined, a possible way to assess the consistency is the ratio between the number of times that the rules are not violated and the number times rules have been checked.

Timeliness In [20] Timeliness has been defined as the extent to which the age of data is appropriate for the task at hand. It refers to the temporal validity of data: data should not be out-of-date when used. In [6], timeliness is defined as composed of two main concepts: (i) *currency* is a measure of how old the data are, based on how long ago they were acquired, (ii) *volatility* is the frequency of change of a value. To make it simple, let us consider reading a value v in a certain time instant, it is possible to define the currency as the number of days from the last update and the volatility as the average number of days for which the value is considered stable (i.e., it is unlikely to change). The value can be considered out-of-date if the inverse of the volatility (i.e., the average number of days between changes) is lower than the currency.

3.4 Data quality improvement

When data are characterized by a poor quality level, it is possible to adopt two general types of strategies to improve them, namely data-driven and process-driven. *Data-driven strategies* improve the quality of data by directly modifying the value of data. *Process-driven strategies* improve quality by redesigning the processes that create or modify data. In summary, the goal of data-driven strategies is to correct the error while process-driven strategies aim to find and eliminate the root cause of an error. *Data cleaning* is a data-driven strategy that aims to identify and eliminate inconsistencies, discrepancies, and errors in data to improve quality [18]. Figure 3.2 illustrates the steps to perform to clean a data set. It is possible to notice that, once data have been collected, before starting the cleaning process, *Data profiling* is performed. Data profiling is the process of analyzing the data available from an existing information source (e.g., a database or a file) and collecting statistics or informative summaries about that data [14].

Examples of tasks performed in the data profiling phase are:

- Analysis of content and structure of attributes: identification of the data types, definition of data distribution and variance, number of null values per attribute, number of distinct values per attribute.

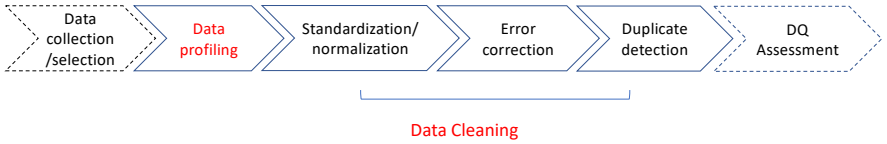


Figure 3.2: Data cleaning process

- Analysis of dependencies between attributes (e.g., Functional dependencies, primary key candidates).
- Calculating statistical data like minimum, maximum, percentile, mean and standard deviation of the numerical values.
- Identification of duplicates.

All the information provided by the data profiling module is used as input for the data cleaning process which is composed of three main steps: standardization/normalization, error correction, and duplicate detection. In the *standardizationnormalization phase*, the goal is to have a homogeneous representation of the values. For each attribute, for example, a specific format is defined (e.g., dates have to be written in the same way) and abbreviations and acronyms can be substituted with the complete expression. The *error correction phase* is focused on the identification and correction of inconsistencies, null values, and outliers.

Duplicate detection is the discovery of multiple representations of the same real-world object. If we have multiple representations in the same data set it means that they are similar. Therefore we need to use specific functions for the evaluation of the similarity. The main issues related to duplicate detection are the identification of a good similarity measure and the minimization of the number of comparisons. At the high level view, the duplicate detection methods compare the tuples of a dataset by calculating the similarity value. Usually, the methods need two thresholds t_1 and t_2 for classifying the tuples as duplicates or not duplicates. If the similarity is greater than t_1 then the tuples represent the same object while if the similarity is lower than t_2 they are referring to two distinct objects. Between t_1 and t_2 there is an uncertain region in which the method is not able to classify the tuples. Usually, in these cases, the duplicate detection tools ask for a human intervention.

3.5 SQL for data profiling and data cleaning

SQL can be used for data profiling, in particular for large databases. In fact, it provides functions for analyzing the data with queries, which are run within the DBMS environment where the data is stored, without the need to transfer all the data before performing the analysis.

For *data profiling*, SQL provides a GROUP BY clause that allows considering groups of values together, and applying to them statistical operations such as COUNT, MIN, MAX, AVG.

For instance, the following SQL statement counts the number of tuples with a given value for an attribute, their Min, Max, and Avg values for all the existing values:

```
SELECT attr_name, COUNT(*), MIN(attr_name), MAX(attr_name),
AVG(attr_name)
FROM Table
GROUP BY attr_name
```

Relations are sets of tuples. Set operators can be applied: UNION, INTERSECT, EXCEPT.

```
SELECT Attr1, Attr2, ..., AttrN
FROM Dataset1
EXCEPT
SELECT Attr1, Attr2, ..., AttrN
FROM Dataset2
```

For *data cleaning*, the following SQL statements can be used.
SELECT DISTINCT RETURNS the result without duplicates:

```
SELECT DISTINCT attr1, attr2, ..., attrn
FROM tables
WHERE condition
```

UPDATE STATEMENT to set a new value (or NULL) for an attribute of a tuple

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition
```

3.6 Exercises

1. **Data Quality Problems (single source) - example**

   Find the quality problems in the following table (Fig. 3.3):

2. **Data Quality problems (multiple sources) - example**

   Find the quality problems in the following table (Fig. 3.4):

3. **Data profiling and cleaning exercises**

   There are several Python libraries to analyze data quality aspects, in particular for profiling attributes. In this exercise, we use the *sweetviz* library as follows:

ID	Name	Street and house number	Postcode	Town	Date of birth	Phone	e-mail
1	Janet Gordon	30 Fruit Street	75201	Dallas			
2	Kathy Robert	436 Devon Park Drive	94105	San Francisco	08.08.1969	215-367-2355	krob@robert.com
3	Sandra Powels	3349 North Ridge Avenue	33706	St. Pete Beach			
4	Johnstone, Jeffrey	3300 Sylvester Rd	92020	El Cajon			
5	Lowe Ruth-Hanna	25 Peachtree Lane	02112	Boston	10.10.50	(0617)-8845123	
6	Gordon Janet	30 Fruit Street	75201	Dallas			
7	NIck Goodman	Regional Campuses, 711	10020	New York	08/07/1975		n.good@goodman.com
8	Poweles Donna S.	3347 North Ridge	33706	Saint Pete Beach			
9	Cathy Robbert	436 Devon Park Drive	94105	San Francisco	08.03.1969		
10	Ruthanna Lowe	25 Peachtree Lane	02112	Boston		0617-8845123	
11	John Smith	10 Main Street	02112	New York			
12	Robert Katrin	434 Devon Park	94105	San Francisco			
13	Nick Goodman	56 Grafton Street	94105	San Francisco	08/07/1975		n.good@goodman.com
14	Sandro Powels	3349 North Ridge Av.	33706	Pete Beach			

Figure 3.3: Data quality problems (single source)

ID	Diagnosis	Hospital	Province	Date	Cost
1	Flu	SR	Milan	01/05/2008	200
2	Flu	SR	Milan	24/5/2008	180-220
3	Flu	SR	Milan	04/05/2008	9999
4	Influenza	SC	Trento	03.05.2008	
5	Influenza	SC	Trento	03.04.2008	230
6	Influenza	SC	Trento	10.07.2008	
7	Flu Type A	CG	Milano	04-04-2008	130
8	Flu	OS	Bolzano	2008/04/23	130
9	Flu	OS	Bolzano	2008/05/11	200

Figure 3.4: Data quality problems (multiple sources)

```
import sweetviz as sv
sweet_report =sv.analyze(df)
sweet_report.show_notebook()
```

The produced report profiles the dataset, and for each attribute it performs an analysis of its values distribution and its completeness.

In the Python notebook at `https://colab.research.google.com/drive/1UYERjJUOAFrcuXLXcd9udQ3ZA7xrtPpA?usp=sharing` some running examples of profiling with sweetwiz and of profiling and cleaning with SQL are illustrated. The used datasets can be downloaded from `https://www.dropbox.com/sh/ww92mqe6fp0vrfo/AABVDAOIHugWHMse-8GmPq6Na?dl=0`

and are in the files property.csv, stock1.csv, stock2.scv.

Chapter 4

Data analysis pipelines

Barbara Pernici, Monica Vitali

4.1 Introduction

As discussed in the first chapter, decision-making is more and more based on data (Data-driven Decision Making DDDM).

To be able to use data for decisions, it is necessary to elaborate the data along different steps. The main ones that have been discussed so far are summarized in Fig. 4.1. In the last chapter, we have focused mainly on data quality issues and on a first introduction of techniques to improve the quality of collected data.

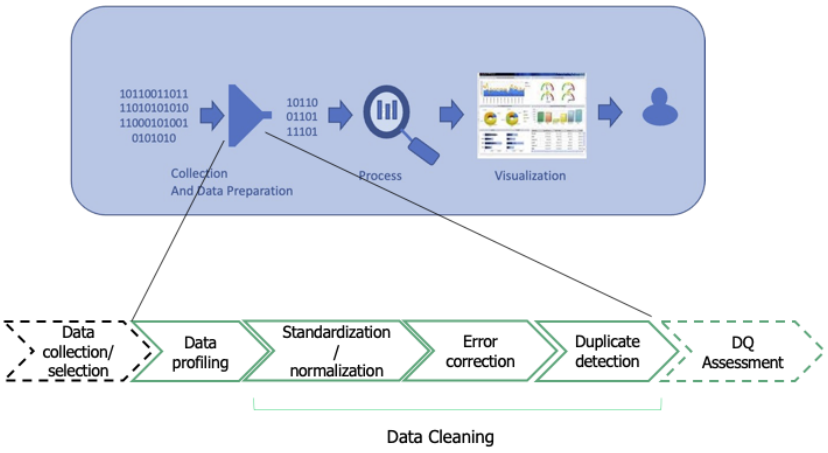


Figure 4.1: Data analysis pipeline

In the present chapter, we discuss different types of pipelines first, then on the characteristics of data sources and types of data.

4.2 Scenarios

We discuss here three main scenarios that require data analysis pipeline.

4.2.1 Datawarehousing

Data is an important resource not only at an operational level but also at a tactical and strategic level to support decision-making processes within an organization. Therefore, the need arises to access aggregate data in real-time and to perform complex queries, such as the analysis of the correlation between different variables (for example, the relationship existing between a customer's place of residence and his or her predisposition to use a certain category of services). As you move from the operational level to the higher levels of Anthony's pyramid, both the type of data of interest and the way it is used change. For tactical and strategic activities to be successful, it is necessary to provide tools capable of extracting information of interest from different sources (internal and external) in a fast and effective way. These tools fall within the so-called *Business Intelligence (BI)* and we are dealing with *informational data* (also called analytical data in the following).

The first decision support tools were reports. Reports contain analytical data, obtained from the analysis of the operational database, which are produced following a fixed format at regular intervals. This tool suffers from several limitations, as the data contained is static data with which the user cannot interact to obtain details or further information on some unexpected aspects. Furthermore, the production of reports can be complex and take a considerable amount of time, making it impossible to access updated information. Finally, the reports give a limited view of the aspects of interest for strategic decisions as they are based only on the internal data of the organization that is extracted and aggregated from the operational database.

More advanced and in many respects more dynamic tools are spreadsheets (e.g., excel tables). In this case, the analyses can be defined directly by the end user who can change them at will and run them when the information is needed. Despite these advantages over reports, spreadsheets are still limited due to the complex procedures required to export the data from the database and import them into the spreadsheet. Furthermore, by using such a tool, each user can create a customized tool which, despite the advantages, does not have the benefit of an integrity and accuracy check of the analysis that is typical of shared tools.

A tool capable of supporting the activities at the highest levels of Anthony's pyramid is instead a database with characteristics different from the classic operational databases:

- The database may include data from operational databases and data from external sources.
- The data must be organized according to a structure that facilitates analysis.

- The structure of the database must be simple and relate all and only the data of interest.
- Data sources need to be integrated and updated.
- Analysis tools with short response times should be available.

To meet these needs, the concept of *datawarehouse* was born (see Fig. 4.2). A datawarehouse allows extracting data from different sources and integrating and aggregating them to provide a technological infrastructure to query, aggregate, and analyze them. The different components will be illustrated in Chapter 8.

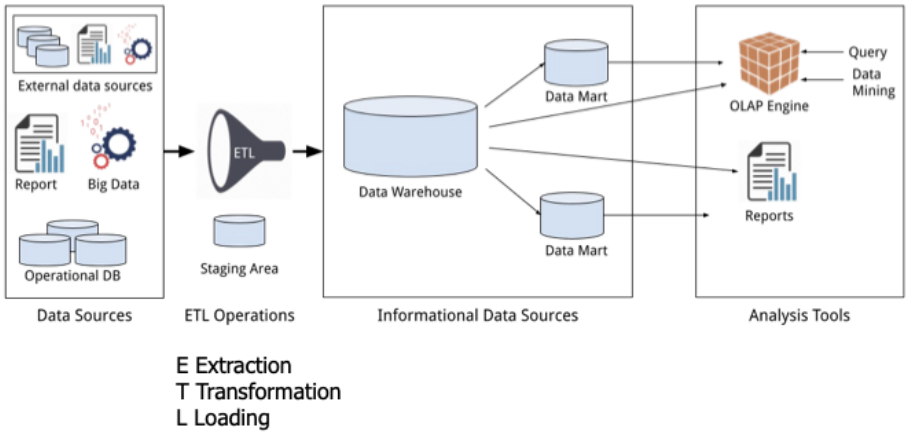


Figure 4.2: Data pipeline in datawarehousing

4.2.2 Data science pipelines

As already discussed in the chapters concerning data quality and data warehouses, to perform data analysis it is necessary to carry out some transformation operations before being able to use them. The growing diffusion of the automated data analysis tools illustrated above led first to the definition of reference architectures for Big Data, as it will be discussed in the Chapter 7, and then to outline the characteristics of the various phases in greater detail and to develop specific products to support activities related to Data Science, with the definition of a life cycle for the main phases related to the analysis process.

As regards a typical data science process, the scheme proposed in [5] is shown in Fig. 4.3, which outlines the activities of the main phases: Acquire, Cleaning / Transformation (Clean), Use/Reuse, Publication, Preserve/ Destroy.

4.2.3 Machine Learning pipelines

"Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way humans learn,

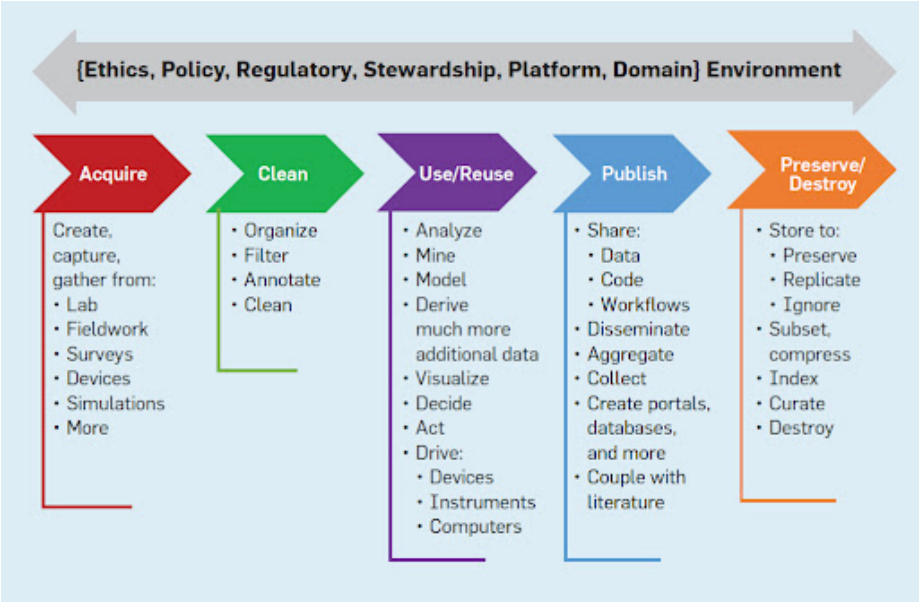


Figure 4.3: Data Science pipeline [5]

gradually improving its accuracy"[1]

The value of Machine Learning lies in the fact that it allows learning from data, predicting future behavior, detecting trends, or understanding patterns. Companies are increasingly adopting this set of algorithms to improve processes and extract knowledge from data patterns and anomalies.

The process of using ML algorithms consists of three steps: data acquisition, generation of the ML model starting from a subset of data called the training set, and use of the model to generate results (Fig. 4.4).

ML models are divided into online and offline. The online models are continuously updated as data is acquired. Offline templates once created remain unchanged over time. The advantage of online models lies in the fact that continuous updating usually corresponds to continuous improvement in the quality of the results.

Machine learning is a form of AI that allows a system to learn from data rather than through explicit programming. Within machine learning techniques, in *supervised learning* the training set must be a "labeled" data source where the label defines the meaning of the data and provides a first idea of how the data can be classified. When the labels are continuous we have regression techniques, when the labels are extracted from a finite set of values we have classification techniques. Regression helps to understand the correlation between variables (e.g., weather forecast). Supervised learning is intended to find data classification patterns that can be applied to an analytics process. In supervised learning

[1] `https://www.ibm.com/cloud/learn/machine-learning`

it is necessary to try to avoid the overfitting phenomenon: the model is capable of representing the training data in a very precise way, but is not able to generalize the observed examples and therefore could be less effective when applied to other sets of data.

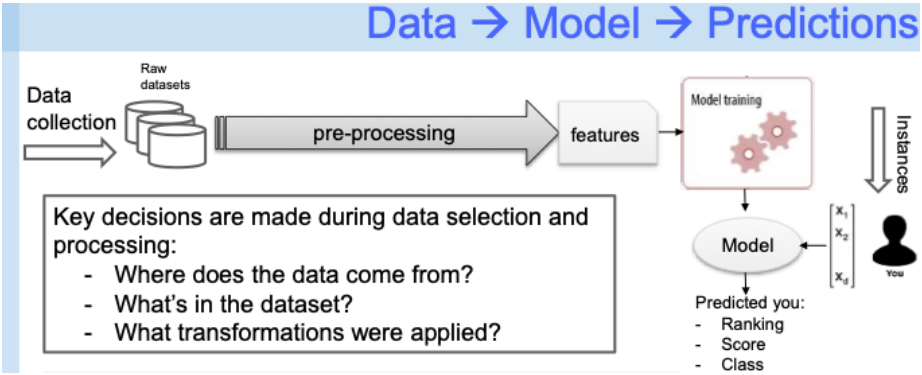


Figure 4.4: Data analysis pipeline (source Missier, 2021)

As for other pipelines, the pre-processing phase is critical also in training ML models. In fact, in addition to data cleaning, the transformations applied to data may influence the obtained model, generating, for instance, biases. Completeness of the training data is also an important factor, as the model is based on the coverage of different situations.

4.3 Data sources

Data can originate from different types of sources. In an organization, usually, we distinguish between internal and external sources.

Internal sources include different internal databases, used in the OLTP systems of the organization, but also data originating from sensors in cyberphysical systems (such as machines, gates, building control, monitoring systems, and so on); we often refer to such data as big data (see Section 4.3.1). In addition, sources of data can be internal documents in a textual form, such as periodic reports, regulations, etc.

External sources include external databases or data repositories, textual documents, and different sources for big data.

Repositories include general ones, such as zenodo[2] or kaggle [3], or repositories of institutional data, e.g., official statistical data from National Statistical Offices (ISTAT[4] in Italy), eurostat[5].

[2] https://zenodo.org/
[3] https://www.kaggle.com/datasets
[4] http://dati.istat.it/
[5] https://ec.europa.eu/eurostat/web/main/data/database

4.3.1 Big data

Big data are massive sets of unstructured/semi-structured data from web traffic, social media, sensors, and so on. Their volumes are too great for typical DBMS, ranging from several terabytes (TB 10^{12} byte), to petabytes (PB 10^{15} bytes), exabytes (EB 10^{18} bytes), Zettabyte(ZB 10^{21} bytes) of data). Usually, an information system of an organization that contains operational level data contains around 1-5 terabytes. Using big data can help reveal more patterns, relationships, and anomalies in many different domains and requires new tools and technologies to manage and analyze them.

Use cases

Some common use cases in which big data insights help improve decision-making, enter new markets, and deliver better customer experiences include: manufacturing, retail, healthcare, oil and gas, telecommunications, and financial services. Some use cases are reported in `https://www.oracle.com/a/ocom/docs/top-22-use-cases-for-big-data.pdf`.

For instance, in the *Manufacturing domain* typical applications include predictive maintenance, improving operational efficiency, and production optimization. The challenges encountered in this domain include data integration from sources using *different formats* and the identification of the signals that will lead to optimizing maintenance. It is needed to balance the data volume with the *growing number of sources*, users, and applications. The analysis includes production equipment data, material use, and other factors, and combining the *different kinds of data* can pose a challenge.

Big data characteristics

Big data have been associated with several characteristics, often cited as the V's of big data[6]. The main ones are the following:

- *Volumes*. The volumes of data being produced are increasing exponentially, with an estimation of 120 zettabytes of data created in 2023 (Fig. 4.5).
- *Velocity*, i.e., the high production rate, typical of streaming data, e.g., sensors that collect numerous data that have to be analyzed. From data to be queried on demand to dynamic data to be analyzed in real-time (Data streaming). "Sometimes 2 minutes is too late. For time-sensitive processes such as catching fraud, big data must be used as it streams into your enterprise to maximize its value. Scrutinizing 5 million trade events created each day to identify potential fraud or analyzing 500 million daily call detail records in real-time to predict customer churn faster"[7] are technically challenging.

[6] `http://www-01.ibm.com/software/data/bigdata/`
[7] source `https://www-01.ibm.com/software/in/data/bigdata/)`

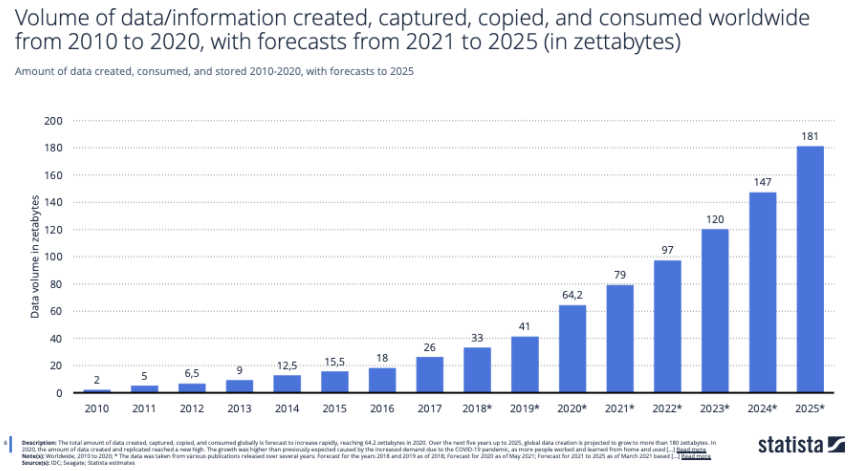


Figure 4.5: Data volumes (source: Statista)

- *Variety*, the different forms of data, including sensor data, videos, databases, posts, tweets, and so on (Fig. 4.6).
- *Veracity*, the data can be uncertain, many business leaders do not trust the information to make decisions. Reasons for uncertainty include: network problems, erroneous data inputs, unskilled operators, unreliable sensors, measuring errors, opinions, faulty image recognition, untrusted sources, etc.

Sources of big data include:

- Conversation text data, e.g., Twitter, Facebook
- Photo and video, image data, e.g., Youtube
- Audio files, e.g., call centers
- Sensor data, e.g., smartphones, geoseismic data
- The Internet of Things data, e.g., smart devices
- Web customer data, e.g., Weblogs
- Traditional customer data, e.g., receipts, loyalty programs, traffic data of telephone/Internet operators

All that content and constant bombardment with messages presents an equally huge challenge for marketers who need to get their message heard through the storm. There's even a term for it - *Content shock* - coined by Mark Schaefer.

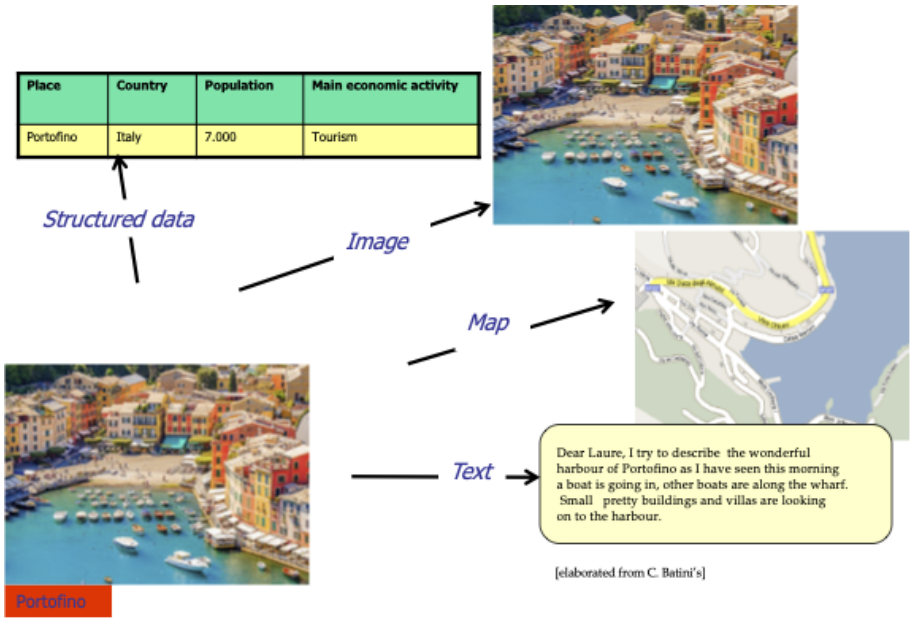


Figure 4.6: Data variety (elaborated from Batini's)

An estimation of the data produced in one minute online is given in Fig. 4.7.

Big Data analysis allows understanding customer needs, improving service quality, and predicting and preventing risks. High-quality data are the precondition for guaranteeing the quality of the results of Big Data analysis.

Big Data tried to overcome Data Quality issues with Data Quantity. But quality is still an issue.

Big data quality challenges include[8]:

- *Diversity of data sources* (*Variety*):
  - Abundant data types - internal + external data sources
  - Complex data structures - structured, semi-structured, IoT
  - Difficult data integration - ETL and traditional approaches useless due to data volume and velocity
- *Tremendous data volume* (*Volume*)

  Data quality profiling and assessment (collection, cleaning, and integration) is difficult to execute in a reasonable amount of time.
- *Timeliness of data is very short* (*Velocity*)

[8]source Cai, Li, and Yangyong Zhu. "The challenges of data quality and data quality assessment in the big data era." Data Science Journal 14 (2015)

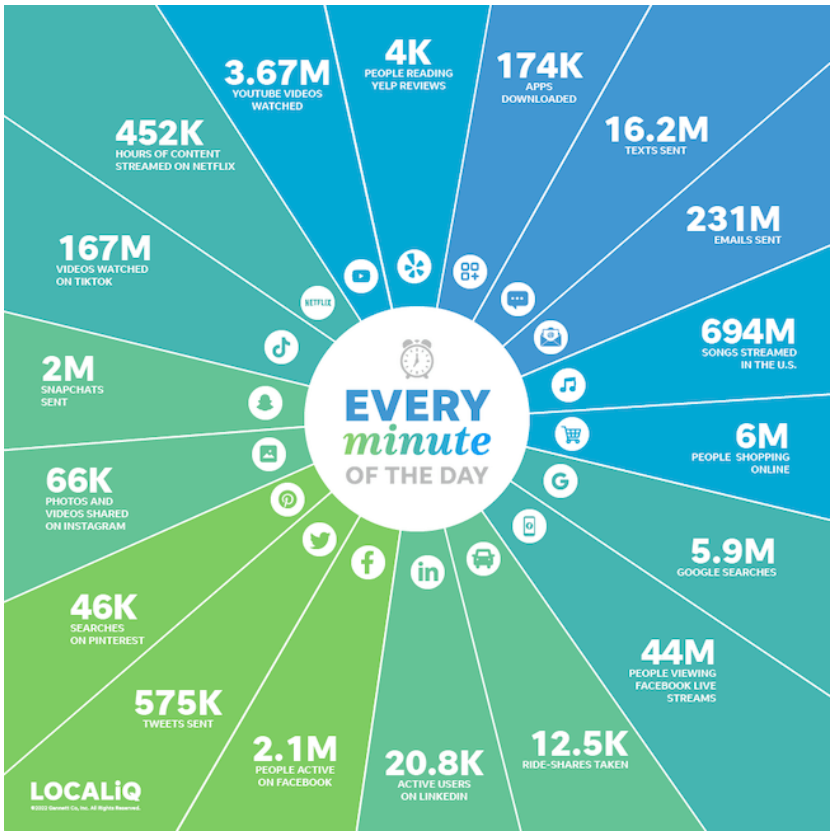


Figure 4.7: Data produced online in one minute (source LocaliQ 2022)

Data is updated continuously. If data is not collected and analyzed in real-time, information becomes outdated and invalid.

- *Missing standard for Data Quality* (*Veracity*)

  Standards have been proposed for DQ of traditional data sources but not for big data.

4.3.2 Source selection

Different sources may be available to choose from. Data can be collected from one or more sources.

Selection criteria:

- Accessibility (internal, external), licensing
- Quality of data, in particular Accuracy, Completeness, Timeliness (real-time updates)

- Availability of metadata, simplicity of integration
- Reputation of source

  The reputation of the source can be assessed by considering how often a dataset is reused, and based on trust in the provider (e.g., participating in a Data ecosystem)
- Cost

4.4 Data integration

Data from different sources may have characteristics that do not make them compatible with each other. For this reason, they must be transformed because it is not certain that application modules, while handling the same data, use them represented according to the same format. As mentioned in the previous chapter, the integration problems concern both the *format* and the *semantics* of the data.

Let us first consider the *format* of the data, which describes how the data is saved and presented. Data format integration issues concern:

- *Representation*: different modules can represent the same information differently. Let us take an address as an example: it can be represented as a record < street / square, house number, postcode, city >, or as a simple text string containing the same elements.

  The representation of addresses may vary from country to country, but also within the same country (think for example of the various names existing in Italy to indicate a street, such as "campo", "calle", etc.). Therefore, representation using a single string allows you to manage addresses in a more general way. However, in this case, there is the problem of recognizing the various fields of the record within the application (for example, separating postcode and city).
- *Structure*: we have the same representation, but the fields are ordered differently. For example, the “address” record could be made up of the same fields, but in different order <ZIP, city, street / square, house number>, or structured hierarchically, for example,
  << Postal code, city>, < street/square, house number>>.
- *Presentation*: the same data can be presented to the user in different formats. For example, a 15-character “surname” field might not be long enough to represent all possible surnames. Therefore, different application modules can present the surname with fields of 15, 20, 30 characters. The correspondence between the strings of different information systems is not guaranteed if the surname exceeds the minimum expected length.

Another integration problem arises if we examine the *semantics of data*, which concerns the meaning attributed to information. For example, if we assume we

are managing orders and we have a "price" field, we can associate a numeric value to it. In the case of an Italian information system, this value can represent a price in euros. However, this interpretation is not general. If we have to integrate the information systems of a company in Italy and a company in Great Britain, in addition to the value, the currency must also be indicated. Some assumptions will also be made on how to convert between one currency and another: will the conversion be done at the time of the order or at the time of payment? By whom will the exchange rate be provided? The problem mentioned above may arise not only concerning currencies, but also for the units of measurement considered: am I ordering 10 kilos or 10 pounds of a product?

The integration between information systems, both between companies, as well as within companies operating internationally, will often present problems of this type.

4.4.1 Data fusion

In addition to the representation, structure, and presentation issues presented in the previous section, when integrating data from different sources it is important to identify data referring to the same entities, before merging the datasets. As discussed within data quality, this requires the ability to compare data from different sources and decide if they are similar enough to be considered data about the same entity. Usually for this operation similarity measures are defined and thresholds set to assess if two elements are very likely the same or they are likely to be different. If conflicts arise or the identification is unsure, these issues must be solved to be able to use the data.

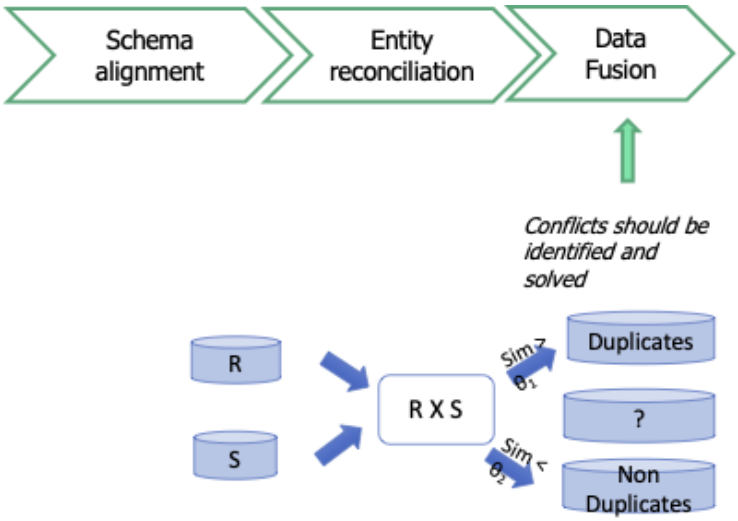


Figure 4.8: Towards data fusion

This process usually is performed in the three phases shown in Fig. 4.8:

i) *schema alignment*, in which the data integration problems mentioned above are solved; ii) *entity reconciliation* to identify data about the same entities and solve the problem; finally, iii) *data fusion* can be performed.

4.5 Tools to support pipelines

The growing diffusion of the automated data analysis tools illustrated above first led to the definition of reference architectures for Big Data and subsequently to an increasingly detailed outline of the characteristics of the various phases and to the development of specific products to support activities related to Data Science, with the definition of a life cycle for the main phases related to the analysis process.

Pipelines require several technological components to be realized.

Often the individual components are supplied by different suppliers, as illustrated in Fig. 4.9, which illustrates some of the tools available on the market. Regarding what is illustrated in the course, notice that data warehouses can be used to store data for analysis, but also other forms of massive data storage, called Data Lakes, characterized by the fact that data storage does not require their a priori integration as in data warehouses, and they allow for the integration of hybrid data in real-time according to needs.

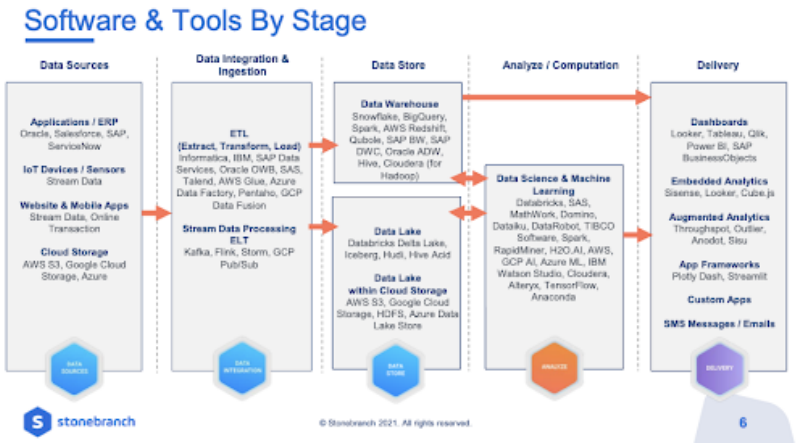


Figure 4.9: Pipeline tools - source Stonebranch, do not redistribute

Chapter 5

Data formats and standardization

5.1 Introduction

As mentioned already before, data can have a well-defined structure, as in relational databases, however in many cases this structure is variable over time, or it is semi-structured, or also we can have unstructured data.

In this chapter, we consider the case of data in which the structure is variable and semi-structured, discussing standardization directions.

We recall how to present structured data in a relational table, for instance, information about a person (Fig. 5.1).

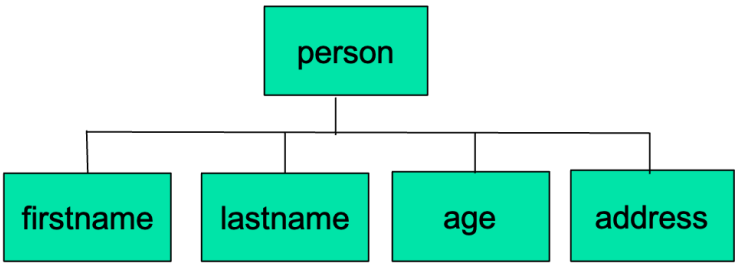


Figure 5.1: Structured data - example

In this case, a relation schema can be defined as follows:

```
PERSON(firstname, lastname, age, address)
```

We recall here that the "address" attribute, in this case, is defined as a string, so we are not providing details of the address, such as town and ZIP, separately, but we will need to extract this information by analyzing the text string. In case

we want to give a structure to addresses, there is the risk of having structures suited for some countries, but not for others.

Therefore, we want to address the case in which some structure exists, however, it can be variable according to the information we want to represent.

In subsequent sections, we will also analyze the problems arising from other types of data, such as free text and images, and how they can be represented to be able to analyze the information they contain.

Problems to be addressed We want to focus in this section on some issues we already encountered, in particular when discussing the quality of data:

- *Multiple values for an attribute.* Cases such as multiple values for a given attribute are not easily represented in relation tables nor in spreadsheets (e.g., more than one telephone number associated with a person).
- *Hierarchical structures and nested structures.* In this case, we would like to give a structure to an attribute, however, this structure may be variable, e.g., for a sensor, we want to represent the time of the collected value, and a value, but this value can change depending for instance on the current configuration of the sensor (e.g., it can be configured to transmit temperature or humidity).
- *Variable structures.* Some new attributes may be needed, so some data might have them and others not (e.g., number of pregnancies for a person).
- *"labels".* In many cases we want to associate labels to data, to describe its properties (e.g., point-of-interest, restaurant, and so on for a location on a map). Labels can vary in time according to the requirements of the users.
- *Interoperability.* The value of data is in their repeated use. Therefore one of the goals is the ability to easily exchange data, including also some information about their contents. It is also important that the data is readable by human actors, and not only by systems, who can therefore understand and use the information directly.
- *Data and metadata.* We need to represent both data and metadata. As we saw in the relational model, metadata can be associated with schema descriptions, however, they are not limited to this, and we need to include information about the source of data, when it was created, measurement units, and the like.

5.2 Logical data models characteristics

We introduce first a generic data model at the logical level (as we did for relational data), which allows for representing some of the issues mentioned above.

5.2.1 Multiple values for an element

In Fig. 5.2, we show a representation of the case in which multiple values for an attribute are possible.

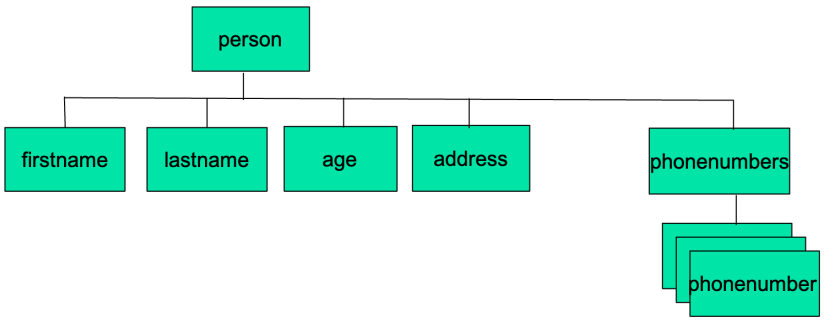


Figure 5.2: Example of multiple values for an element

In a relational database, to represent this information we need two relation tables (here for simplicity in the discussion we assume that firstname and lastname identify a person):

```
PERSON(firstname, lastname, age, address)
PHONENUMBER(firstname, lastname, phonenumber)
```

This implies a JOIN operation every time we need to retrieve information about a person, which is a costly operation, as the data is separated into different tables. The readability of the information is also difficult.

5.2.2 Hierarchical structures

An attribute that can have components can be represented with a hierarchical structure. For instance, in Fig. 5.3 we have a representation of a possible structure for addresses, as a nested structure composed of streetaddress, city, state, postalcode.

Hierarchical structures are not supported by relational models.

5.2.3 Variable attributes and labels

For an entity, its elements may not all be known in advance, at design time, and the need for new terms may emerge. This is a common situation when labels are associated with objects, to add information as they are acquired.

A solution is to allow labeling with so-called <**name, value**> pairs, where the name is used to identify the property and value is used to assign a value to it, e.g., "color"="blue". If new properties emerge, new labels can be created, e.g., "size"="small".

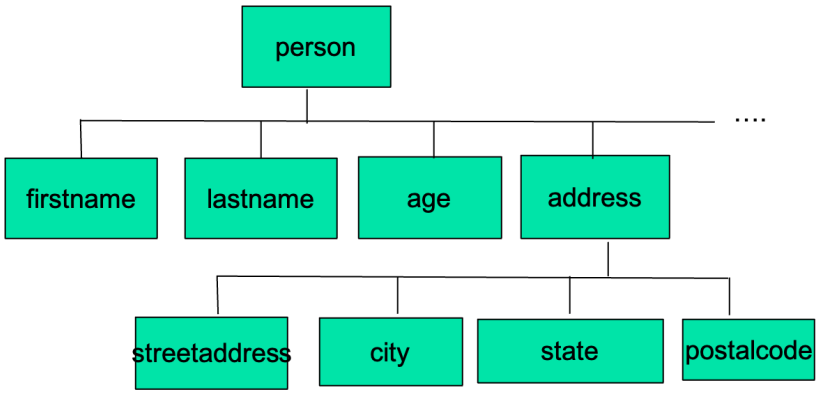


Figure 5.3: Example of hierarchical structure

5.2.4 Structures

In the following, we illustrate two of the most commonly used data formats used for data with variable structures and unstructured data: JSON and XML.

In both cases, we have the following common characteristics:

- Schema: a schema is provided also for semi-structured data, indicating which are the possible elements.
- Data are structured according to a schema, but variable, nested structures are allowed, as well as multiple values. Data are provided in a textual form, that can be read in programs, e.g. in the case of Python importing the "json" package.

These representations can also be natively stored in some DBMS allowing these formats and can be used as Interchange formats for interoperability.

5.3 JSON – JavaScript Object Notation

5.3.1 Introduction to the JSON notation

JSON is based on “name”: “value” pairs to represent values and their meaning, separated by commas:

```
"name1": "value1",
"name2": "value2"
```

Nesting is obtained through parentheses { }, creating hierarchical hierarchical structures. This unit of information is called *object* and it is a collection of information items,

Elements in a list used to represent multiple values are represented between square brackets [ ].

A list of data items is represented as follows:

```
[{"name1":"value11"}, {"name1":"value12"}, {"name1":"value13"}]
```

In this case, three "name1" values are provided for an item.

In Fig. 5.4, we represent person data in JSON, with the structure illustrated before:

```
{
  "person": {
    "firstName": "John",
    "lastName": "Smith",
    "age": "25",
    "address": {
      "streetAddress": "21 2nd Street",
      "city": "New York",
      "state": "NY",
      "postalCode": "10021"
    },
    "phoneNumbers": {
      "phoneNumber": [
        {
          "type": "home",
          "number": "212 555-1234"
        },
        {
          "type": "fax",
          "number": "646 555-4567"
        }
      ]
    }
  }
}
```

Figure 5.4: Example of JSON for person

In this case, we have an object person, with its information items: "firstName", "lastName", "age", "address", "phoneNumbers". The item "address" is in turn an object with four data items: "streetAddress", "city", "state", "postalCode", while "phoneNumbers" is a list containing two items "phoneNumber".

Metadata What about metadata? there is no distinction in JSON, which does not have the concept of attribute for objects like in XML.

5.3.2 GeoJSON

GeoJSON is a format for encoding a variety of geographic data structures proposed in an open-source format by Internet Engineering Task Force (IETF). *Feature objects* in geoJSON have a predefined type *Feature* and have a *geometry*, which indicates the position of the object in terms of one or more longitude and latitude coordinates, and one or more *properties*.

For instance, Duomo square in Milan can be indicated as follows[1]:

```
{
  "type": "Feature",
  "properties": {
    "display_name": "Cathedral Square, Duomo, Municipio 1, Milan,
    Lombardy, Italy",
    "category": "place",
    "type": "square"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      9.190662109391159,
      45.46406575
    ]
  }
}
```

GeoJSON supports the following *Geometry types*: Point, LineString, Polygon, MultiPoint, MultiLineString, and MultiPolygon. You can try to draw different shapes on a map using `http://geojson.io/`, an interactive tool that allows displaying or creating points on a map, and importing and exporting geographical information in geoJSON. It has to be noted that in geoJSON "properties" is a reserved word, however, properties can freely be added by the user (OSM has some internal rules for adding properties[2]

Geometric objects with additional properties are *Feature objects*. Sets of features are contained by FeatureCollection objects.

Here is an example of a collection of points on a map, which is displayed using geoJSON.io in Fig. 5.5.

[1]This information is derived from OpenStreetMap, `https://www.openstreetmap.org/` a map created as a Volunteered Geographical initiative, using the search engine Nominatim, with the search API `https://nominatim.openstreetmap.org/search?q=cathedral, Milan&format=geojson`. Nominatim has also an interactive interface `https://nominatim.openstreetmap.org/ui/search.htmltosearchforgeographicalobjectsinOpenStreetMap`

[2]As an example, a list of predefined categories is available at `https://wiki.openstreetmap.org/wiki/OpenStreetBrowser/Category_list`

```
{
  "type": "FeatureCollection",
  "licence": "Data © OpenStreetMap contributors, ODbL 1.0.
  https://osm.org/copyright",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "place_id": 107700615,
        "osm_type": "way",
        "osm_id": 27626748,
        "display_name": "Milan Cathedral, Piazza del Duomo,
        Duomo, Municipio 1, Milan,
        Lombardy, 20122, Italy",
        "place_rank": 30,
        "category": "amenity",
        "type": "place_of_worship"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          9.191621109614111,
          45.46416835
        ]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "place_id": 204435173,
        "osm_type": "way",
        "osm_id": 463529462,
        "display_name": "Cathedral Square, Duomo, Municipio 1, Milan, Lombardy, Italy",
        "place_rank": 25,
        "category": "place",
        "type": "square"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          9.190662109391159,
          45.46406575
        ]
      }
    }
  ]
```

```
}
```

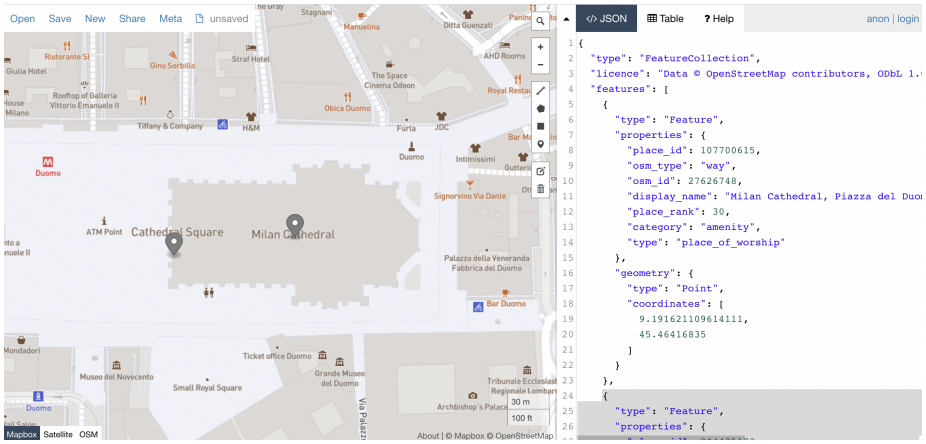


Figure 5.5: Two points displayed on a map using geoJSON.io

5.4 NoSQL DBMS

To store data with semi-structured information, database technology has evolved introducing the term *Not-only relational databases* ("NoSQL").

These new DBMSs allow using flexible data models and usually allow handling large volumes of unstructured and structured data (and documents), with data sets stored across distributed machines, and are easier to scale (scalability is needed when the amount of data increases over time and more resources are needed).

Many relational DBMS have NoSQL extensions, e.g. Postgres.

JSON or JSON-like documents DBMS are MongoDB and CouchDB.

Some DBMS are specialized in handling time series, such as with PostgresSQL extensions and Graphite. Big companies such as Booking.com, Reddit, and GitHub use Graphite daily to be able to easily detect outages on their architecture.

XML databases (including document-oriented databases) that support the ISO XML Type are IBM DB2, Microsoft SQL Server, Oracle Database, PostgreSQL. Some native XML databases are also available, such as BaseX.

A comparison of databases is provided in `https://db-engines.com/`. The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly: `https://db-engines.com/en/ranking`

5.5 XML - eXtensible Markup Language

5.5.1 Introduction to the XML notation

In Markup languages, data values are represented together with a tag to understand their meaning.

Therefore, *tags* give a name to data, e.g., person, firstname, ... in the examples above.

In this section, we introduce the *Extensible Markup Language* (XML) text format, defined as an open format by the W3C (World Wide Web Consortium, an "international community that develops open standards to ensure the long-term growth of the Web"), for creating documents that are both human-readable and machine-readable.

In XML, open and close clauses are used to delimit a data item within a given tag as follows:

<tag>... data ... </tag>

For instance:

<firstname>Anna</firstname>

Tags can be nested (e.g., to represent the hierarchical data structure above). In addition, different values for the same logical component can be provided, repeating the tag of the component.

An example is shown in Fig. 5.6 to represent person data in XML. Please note spaces and indents do not carry a semantic meaning. A compact form can be used for the same data.

5.5.2 Namespaces

For Markup languages, a catalog of tags can be created using the concept of namespace, i.e., the definition of terms within a given environment. Examples of namespaces are illustrated in Fig. 5.7. Each name can refer to a namespace name and to a local name. For instance, in a filename, we can split the file name in the name of its directory (e.g. /home/user) and the name of the file in the directory, eg., readme.txt.

Namespaces can be internal to a company, to create a repository for all names (tags) used in company data, or they can be shared in standardized namespaces. An example of such a namespace for mathematical formulas is available from W3C: `https://www.w3.org/TR/MathML3/`

5.5.3 Attributes in XML

Within a tag, we can add name-value pairs, e.g. id="123"

```
<tag name1="value1" name2="value2" ....>
... Data ....
</tag>
```

```
<person>
  <firstName>John</firstName>
  <lastName>Smith</lastName>
  <age>25</age>
  <address>
    <streetAddress>21 2nd Street</streetAddress>
    <city>New York</city>
    <state>NY</state>
    <postalCode>10021</postalCode>
  </address>
<phoneNumbers>
    <phoneNumber>
      <type>home</type>
      <number>212 555-1234</number>
    </phoneNumber>
    <phoneNumber>
      <type>fax</type>
      <number>646 555-4567</number>
    </phoneNumber>
  </phoneNumbers>
</person>
```

Figure 5.6: Example of XML for person

The properties can also be serialized using attributes instead of tags:

```
<person firstName="John" lastName="Smith" age="25">
  <address streetAddress="21 2nd Street" city="New York" state="NY" postalCode="10021"
  <phoneNumbers>
     <phoneNumber type="home" number="212 555-1234"/>
     <phoneNumber type="fax"  number="646 555-4567"/>
  </phoneNumbers>
</person>
```

Attributes cannot be nested, so they cannot have a complex structure.

Attributes are most frequently used for metadata, e.g., id, location of a sensor sending time series.

A suggestion is not to put all the values as attributes as in the example shown above, as in this case data will not be separated from metadata and it would be more difficult to retrieve and analyze them.

Context	Name	Namespace name	Local name
Path	/home/user/readme.txt	/home/user (directory)	readme.txt (file name)
Domain name	www.example.com	example.com (domain name)	www (leaf domain name)
XML	xmlns:xhtml="http://www.w3.org/1999/xhtml" <xhtml:body>	xhtml (previously declared XML namespace xhtml="http://www.w3.org/1999/xhtml")	body (element)

Figure 5.7: Examples of namespaces

5.5.4 Pros and cons

Now we compare XML and JSON according to their main characteristics:

- Both are "human readable"
- Both support managing structures that vary in time
- XML has several transformations and query tools
- XML helps distinguish data and metadata
- JSON is more compact
- JSON is easy to use in Python

5.6 Learning resources

- editing and prettyprinting JSON code `https://jsonformatter.org/json-pretty-print/`
- GeoJSON interactive interface `http://geojson.io/`
- OpenStreetMap OSM interactive interface

  `https://nominatim.openstreetmap.org/ui/search.html`

  example of API returning geoJSON `https://nominatim.openstreetmap.org/search?city=Milano&format=geojson`
- Copernicus Emergency Management Services (EMS) - Rapid mapping activations `https://emergency.copernicus.eu/mapping/list-of-activations-rapid` Selecting an event in Details and Downloads you can find json files that can be examined with the GeoJSON interactive interface or used in data analysis

- W3C tutorial on XML: `https://www.w3schools.com/xml`
- XML formatter `https://jsonformatter.org/xml-formatter`
- XML validator `https://codebeautify.org/xmlvalidator`

Chapter 6

Smart objects and time series

6.1 Introduction to the main concepts

At the basis of many applications, the collection of data is performed directly from the environment, often with stringent time constraints. In general, with the term IoT (Internet of Things) we denote objects that can transmit data, with the ability to access the Internet to transmit information where it is eventually processed, for monitoring purposes, to analyze data, or to predict possible future malfunctioning or systems evolution. In general, we assume that objects are associated with some location information, and, in some applications, the position of the object may vary in time.

Transmitted data may vary from a simple identifier for the object to data collected from sensors associated with the object (e.g., temperature, humidity, water levels).

Several technical challenges are posed in the collection of these data, their quality, and making them available to applications.

In this chapter, we focus on three main aspects:

- Object identification technology
- Localization technology
- Time series

6.2 Identification of objects

Associating an id to an object is a recurring problem. Several identifiers are commonly used, for instance:

- License Plates for cars

- Social Security Numbers / Codice Fiscale for persons
- Student ID
- Serial Numbers
- Car Keys codes
- Database Keys
- Bar Codes

Usually, as already examined in Chapter 2, such identifiers are used in databases as keys, and they have a context of validity (e.g., Student IDs are usually defined within a single university). Numeric and alphanumeric codes are often used for this purpose. While textual identifiers may be printed and possibly read with OCR (Optical Character Recognition) tools, bar codes are numerical codes that are designed to improve machine readability.

In addition to identifiers, also locations can be used to identify a fixed object, e.g. a point on a map (i.e., the coordinates of a fixed object) can identify a house or a streetlamp.

In general, some information is associated with an object, e.g., the identity of the owner of a car, the state of a device (on/off), and so on. We distinguish between *static* and *dynamic information* about the object. Static information is fixed, or it changes rarely in a controlled environment, such as the ownership of a car, while dynamic information is subject to change.

Some dynamic information may be available at any point in time, while in other cases it can be gathered only when the object is detected in a given position.

Let us consider for instance a car passing by a toll gate, with an automatic payment system. Usually, the plate is recognized with image processing techniques, and, at the same time, a device (such as, for instance, Telepass in Italy) is recognized by a detector. The position of the car can be inferred by the location of the detector.

The general scenario for such an automatic identification, shown in Fig. 6.1, is based on transmitters and receivers, where the Telepass device transmits an identification code that is recognized by a receiver. To this purpose a wireless network connection is needed for the transmission and the Telepass transmits an identification code. The information about the car, the owner, how the fee is paid and other needed information is managed in an information system storing this information for the fee management system. The information system contains also the association of the Telepass id code with the corresponding user information. The position of a toll booth is known and associated with a location so that the fee can be computed based on entry and exit information as well as the type of vehicle and contract.

In this example, the car is not aware of its position, and its identification by a receiver is used as the basis for the application. Other possibilities for localization will be considered later in Section 6.4.

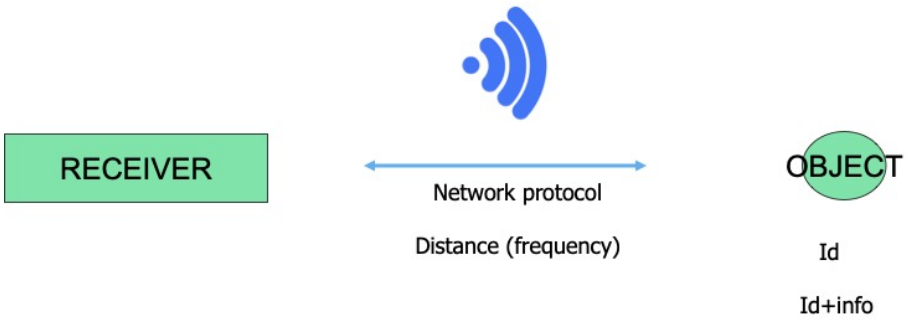


Figure 6.1: An object transmitting its identifier

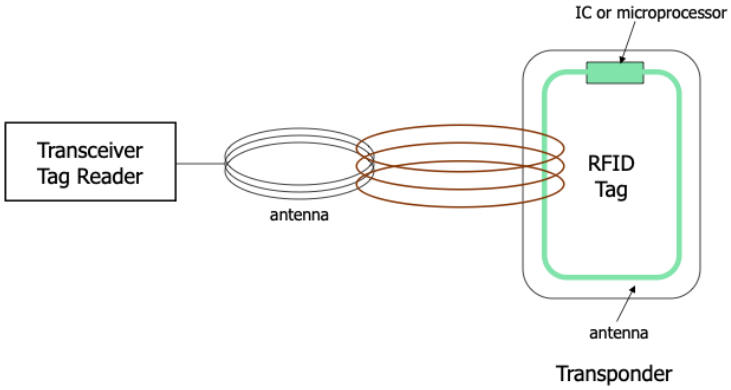


Figure 6.2: Tranceiver and RFID transponder

6.3 Radio Frequency Identifiers

RFID (Radio Frequency Identification) is a technique that allows the recognition of objects electronically, without the need to read their id or barcode with optical recognition.

Radio Frequency Identification uses radio frequency tags to identify real objects, providing an ID for the object.

RFIDs can be passive or active devices.
Passive tags have no battery and low costs.
Active Tags have a battery and a longer range of reach. The disadvantage is the need to replace the battery and higher costs.

RFIDs can have a computational capability, with an associated microchip, or just be an electronic circuit.

The basic mechanism of a passive RFID is illustrated in Fig. 6.2.

Three components of the system can be seen in the figure:

Figure 6.3: RFID

- Transceiver – Tag Reader
- Transponder – RFID tag
- Antennas on both sides

A passive RFID tag does not have its power source, rather the transmission is initiated when the tag is near the reading device (Transponder), which activates the electric circuit of the device by induction.

Passive tags are small in size, lighter, and less expensive, with a shorter read range than active tags. They can be embedded in a sticker or under the skin.

An example of very low-cost RFID, usually found in clothes, is shown in Fig. 6.3.

Active tags are powered by an internal battery and therefore have a longer read range. They are more reliable than passive tags due to the ability of active tags to conduct a "session" with a reader. They have a memory size of up to 1 MB, a bigger size, are more expensive, and have a limited operational lifetime.

An RFID can also have the capability of processing information (e.g., used for encoding data for security reasons), and can also transmit other data about sensors associated with the object.

Tags can be Read only or Read/Write tags.

Read Only tags are factory programmed and usually chipless.

Read / Write tags have an on-board memory, can save data can change ID, and higher cost.

RFID tags are used in a variety of applications. *Supply Chain* RFID tags are usually simple, cheap, no support for cryptography, and provide a single identifier.

Passport RFID tags have shorter intended read range, are tamper resistant, and support cryptography (see Section 10.1.7).

6.3.1 NFC Near Field Communication

Like other "proximity card" technologies, NFC employs electromagnetic induction between two loop antennas when NFC-enabled devices - for example, a

smartphone and a printer— exchange information, operating within the globally available unlicensed radio frequency ISM band of 13.56 MHz on ISO/IEC 18000-3 air interface at rates ranging from 106 to 424 kbit/s. The range is around 10 cm or less.

Each full NFC device can work in three modes:

- *NFC card emulation* enables NFC-enabled devices such as smartphones to act like smart cards, allowing users to perform transactions such as payment or ticketing.
- *NFC reader/writer* enables NFC-enabled devices to read information stored on inexpensive NFC tags embedded in labels or smart posters.
- *NFC peer-to-peer* enables two NFC-enabled devices to communicate with each other to exchange information in an ad hoc fashion.

NFC tags are passive data stores that can be read, and under some circumstances written to, by an NFC device.

NFC technology is often used within smartphones for payment services but can be embedded in any type of object (e.g., rings, bracelets).

6.4 Localization of objects

6.4.1 Localization techniques

After having set the scenario for applications with mobile objects and discussed their identification, we discuss in the next sections how objects can be localized. We focus on the following types of localization:

- with transceivers
- Outdoors:
  - GPS: Global Positioning System

  Indoors:
  - Beacons
  - QR codes

Transceivers

In case transceivers are located in a fixed position, their location can be used to track objects when they are detected.

We were already discussing toll booths, which can record when a given car is passing a given point.

Another common use is in logistics, to track parcels without scanning barcodes. In this case, multiple tags readers can read several tags at the same time can be used.

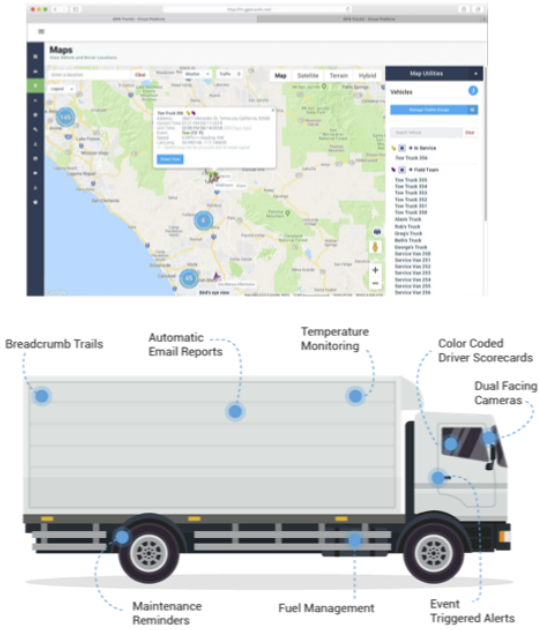


Figure 6.4: Fleet management services

6.4.2 GPS: Global Positioning System

GPS is a satellite-based positioning system (US). The position is calculated by triangulation from signals of at least three satellites.

This technology can be used outdoors and is used, for instance, in many vehicle tracking applications.

Mobile computing is mainly enabled by two distinct technology developments: i) the development of mobile computing devices and ii) the development of wireless networks. This has caused a disconnection of computing and internet access from fixed workplaces, giving way to the always-and-everywhere-connected paradigm. The mobile digital platform can be based on smartphones, Tablets (iPad), and Networked e-readers (Kindle).

An example are fleet management services, like those provided by Verizon[1] Fig. 6.4 shows how the position of a truck and its functioning parameters can be controlled to track the positions of trucks and also recognize possible needs for maintenance and conditions of transportation of perishable goods. Locations are recorded using GPS tracking.

[1] `https://www.verizonconnect.com/solutions/gps-fleet-tracking-software/`

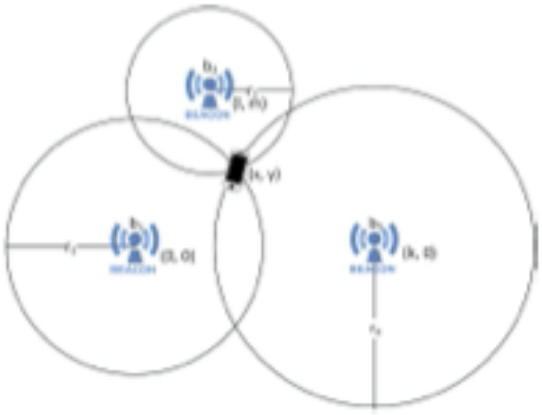


Fig. 1. Example of trilateration with three beacons, b_1, b_2, and b_3 in known locations, (0, 0), (l, m), and (k, 0), respectively, are the transmitters and a smartphone at the intersection, (x, y), as the receiver.

Figure 6.5: Localization with beacons

6.4.3 Beacons

As the GPS system is not performing well indoors, due to the reduced capability of receiving the satellite signal, alternatives are proposed for indoor localization.

Beacons are small, battery-powered, always-on devices that use BLE (Bluetooth Low Energy) technology to transmit signals to devices, such as smartphones and tablets, within a range of about 100 meters. BLE beacons are 1-way hardware transmitters aimed at detecting nearby devices to send them messages, while the target devices do not send information back to the beacons. Beacons transmit a universally unique identifier as a tags' regular signal.

Bluetooth Low Energy is a wireless personal area network technology used for transmitting data over short distances and it is designed for low energy consumption and cost. The advantage of using BLE are: i) Power Consumption: BLE has low energy requirements. It can last up to 3 years on a single coin cell battery. ii) Lower Cost: BLE is 60-80% cheaper than traditional Bluetooth. iii) Application: BLE is ideal for simple applications requiring small periodic transfers of data.

The receiver, usually a smartphone, can detect the signal with the id. More than one beacon is needed for a precise localization (see Fig. 6.5).

Beacons are used in retail, entertainment, museums, industry, and so on[2], to send, for instance, warnings, alarms, and coupons. The general architecture of the system is illustrated in Fig. 6.6.

[2] `https://smartbeacon.it/`

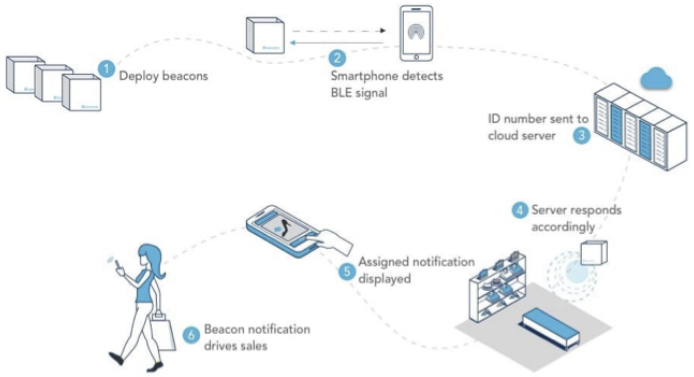


Figure 6.6: How applications with beacons work

Figure 6.7: Sensor data collection (source: http://www.libelium.com/partners-ecosystem/

6.4.4 QR codes

QR codes often contain data for a locator, identifier, or tracker, that points to a website or application

Scanned with a smartphone, it provides a way to access a brand's website more quickly than by manually entering a URL or they can be used as "touchless" system to display information (e.g., in museums to access information about an object on display).

6.5 Time series

Sensor nodes are collecting information that is transmitted at a regular pace or when significant events occur.

There are multiple radio options to communicate through a gateway or directly to the cloud.

Usually, there is a base station that collects data locally, and then transmits them through a gateway to the server analyzing the information, often in the cloud (see Fig. 6.7).

An example from the CTT 2 project is shown in Fig. 6.8.

Figure 6.8: Collecting data in smart cities (CTT 2 Project)

In all these applications, json is often used for transmitting data, due to its flexible structure.

As an example, we present a collection of data gathered at different times.

```
"Object": {
        "type": "Refrigerator Assembly Product",
        "id": "SmartFridge22334411",
        "InfoItem": {
            "name": "Consumed Electrical Power Measure",
            "description": "Power consumption values with timestamp.",
            "value": [
                {
                    "dateTime": "2001-10-26T15:33:21",
                    "Text": 15.5
                }, {
                    "dateTime": "2001-10-26T15:33:50",
                    "Text": 15.7
                }, .....
```

Wrangling time-series data is also known as "garbage in, garbage out", as it can pose many data quality problems. Some examples:

- Timestamping Troubles
- Handling missing data
- Changing the frequency of the time series
- Smoothing data
- Addressing seasonality in data

Time	Intake
Mon, April 7, 11:14:32	pancakes
Mon, April 7, 11:14:32	sandwich
Mon, April 7, 11:14:32	pizza

Figure 6.9: Collecting dietary information (source: Roveri, Falcetta))

Timestamps are quite helpful for time series analysis. From timestamps, we can extrapolate a number of interesting features, such as time of day or day of the week. But.. what process generated the timestamp, how, and when? Often an event happening is not coincident with an event being recorded

For instance, let is consider a sample meal diary from a weight loss app (see Fig. 6.9).

Did the user specify this time or was it automatically created? Does the interface perhaps offer an automatic time that the user can adjust or choose to ignore? Where in the world was it 11:14?

Handling missing data Missing data is surprisingly common in real-world datasets (e.g., communication problems, faults in sensor/actuators, software bugs, etc..). How to deal with that?

- *"Global" filling methods*: When we fill in missing data based on observations about the entire data set.
- *"Local" filling methods:* When we use neighboring data points to estimate the missing value.
- *Deletion of affected time periods*: When we choose not to use time periods that have missing data at all.

Chapter 7

Data Architectures

Barbara Pernici, Edoardo Ramalli

7.1 Reference architectures

7.1.1 Big data pipelines tools and architectures

NIST Big Data Reference Architecture (NDBRA)

The main reference architecture for classifying the architectural components currently available is the one proposed by NIST (National Institute of Standards and Technology, USA) shown in Fig. 7.1, which illustrates, in addition to the modules corresponding to the phases listed above, also the technological infrastructure necessary for Big Data framework providers. Main activities linked to the NBDRA are illustrated in Fig. 7.2.

Big Data Value Architecture

7.1.2 Data lakes

"A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale. You can store your data as-is, without having to first structure the data, and run different types of analytics—from dashboards and visualizations to big data processing, real-time analytics, and machine learning to guide better decisions."[1]

The main components of a data lake and data flows are illustrated in Fig. 7.4.

A functional architecture is shown in Fig. 7.5.

In Fig. 7.6 the characteristics of datawarehouses and data lakes are compared.

[1] https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/

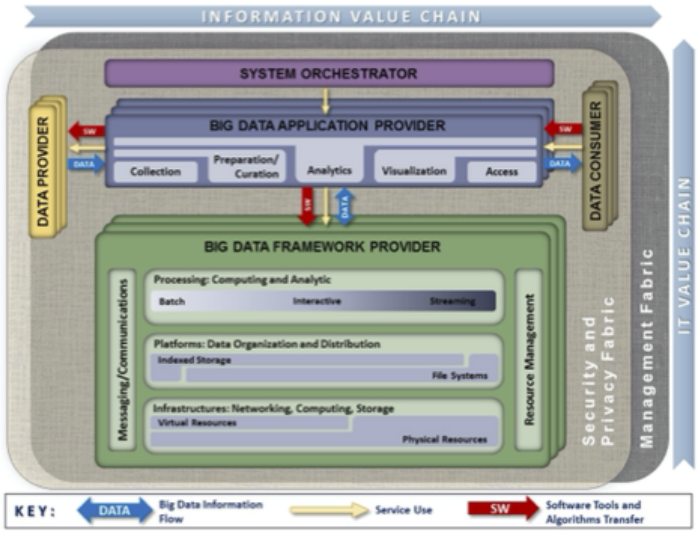


Figure 7.1: NIST Big Data Reference Architecture (NBDRA) [8]

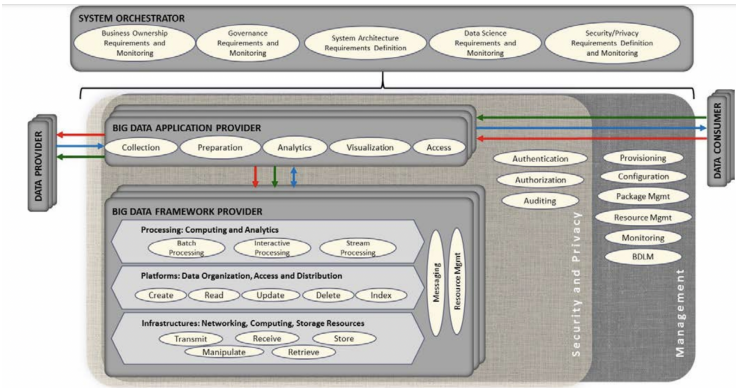


Figure 7.2: NIST Big Data Reference Architecture (NBDRA) -activities[8]

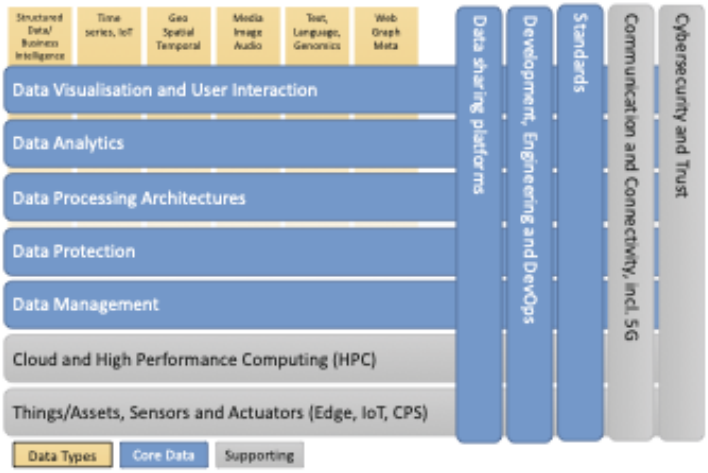


Figure 7.3: Big Data Value Architecture[9]

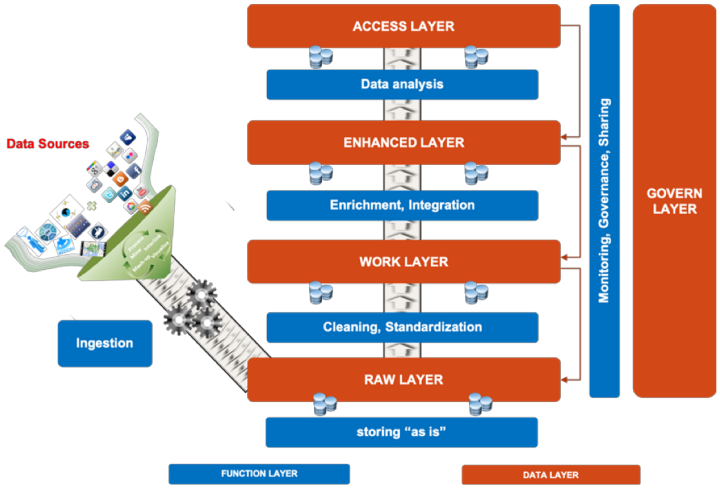


Figure 7.4: Data Lake components [1]

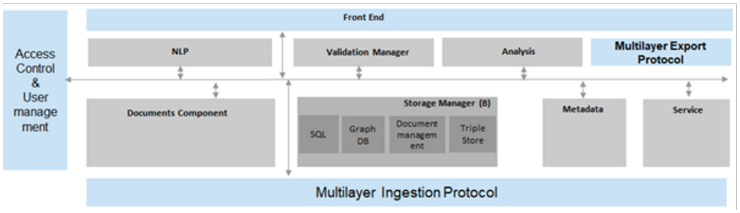


Figure 7.5: Data Lake components [1]

Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

Figure 7.6: Data warehouses and Data lakes comparison`https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/`

7.2 Metadata

example of metadata

`https://ec.europa.eu/eurostat/cache/metadata/en/prc_hpi_inx_esms.htm`
`https://unstats.un.org/sdgs/unsdg`
`http://data.un.org/`
`https://nominatim.openstreetmap.org/ui/search.html?q=milano`

catalogs

7.2.1 Naming styles

As illustrated above, names for properties of elements are often added to existing ones. It is important to keep a consistent naming style when defining new labels.

The most common styles are defined below (note that upper case and lower case may be significant in the notation):

Style	Example	Description
Lower case	<firstname>	All letters lower case
Upper case	<FIRSTNAME>	All letters upper case
Underscore	<first_name>	Underscore separates words
Pascal case	<FirstName>	Uppercase first letter in each word
Camel case	<firstName>	Uppercase first letter in each word except the first one

7.2.2 Data and metadata

As we discussed before, metadata have to be associated with data values to make them usable and understandable.

We distinguish here between two types of metadata, which can be represented in different ways in different notations:

- *Schema*: also for variable structures, a schema can be defined, which allows for representing the structure of each data element. As an example, Fig. 7.7, represents a possible structure for personal information.

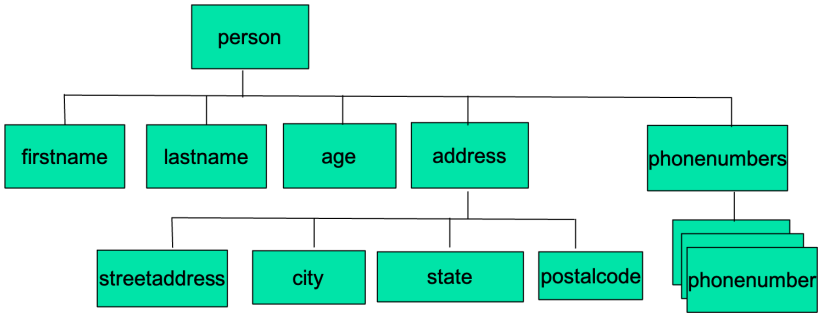


Figure 7.7: Example of schema for person data

- *Other metadata*: data with variable structure may contain values of interest, e.g., stock quotes for given stocks in a given data, but it is also important to keep information such as the source of the data (e.g. Yahoo). Such information is also to be represented, and it can also have a variable structure.

7.3 Data ecosystems

7.3.1 Federated architectures

Two main goals:

- data sharing
- interoperability services

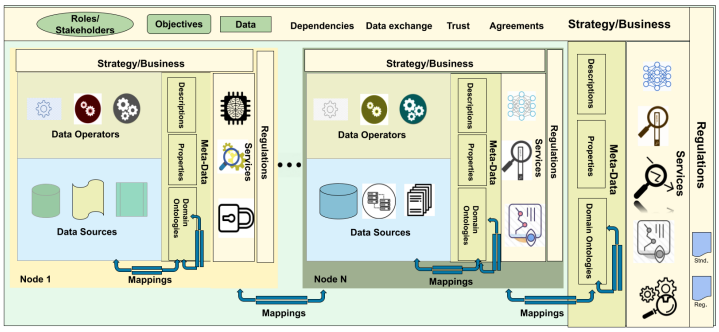


Figure 7.8: Federated Data Ecosystems [13]

A general example of federated environments is illustrated in Fig. 7.8, with its main components.

Several functionalities need to be supported:

- Identity and Trust Services
- Sovereign data exchange
- Federated catalog
- Compliance

7.3.2 Data spaces - Gaia X

"Gaia-X[2] is a project initiated by Europe for Europe and beyond. Representatives from business, politics, and science from Europe and around the globe are working together, hand in hand, to create a federated and secure data infrastructure. Companies and citizens will collate and share data – in such a way that they keep control over them. They should decide what happens to their data and where it is stored, and always retain data sovereignty. The architecture of Gaia-X is based on the principle of decentralization. Gaia-X is the result of a multitude of individual platforms that all follow a common standard – the Gaia-X standard. A data space is a virtual and interoperable system between different cloud service providers, which allows its users to exchange data when needed. Data spaces can be organized by supply chain, sector, or scope."[3]

7.3.3 Provenance

The concept of provenance refers to any information describing the derivation process of any end product. An end product could be an object or digital data. We focus on the latter, even if the concept is general. Thus, provenance tracks

[2] https://gaia-x.eu/

[3] https://www.data-infrastructure.eu/GAIAX/Navigation/EN/Home/home.html

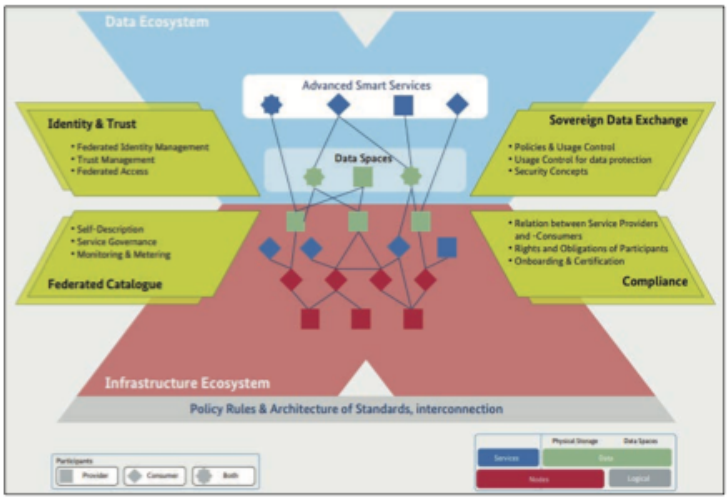


FIGURE 1. High-level view of the GAIA-X architecture.

Figure 7.9: Architecture of Gaia-X [7]

Figure 7.10: GAIA-X Mobility data space [17]

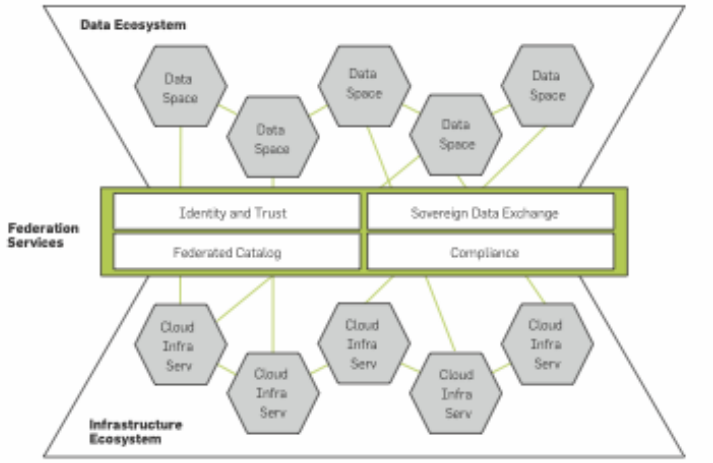


Figure 7.11: GAIA-X ecosystem [17]

metadata about entities, data, processes, activities, and persons involved in the production process of digital data. Essentially, provenance can be seen as metadata that describes a production process instead of describing data. Two examples can help us understand the importance of provenance in different scenarios, such as quality assessment, reproducibility, and enforcing trust in using a product/data. First, in 2013 in the European food market, it was discovered that a product that was supposed to contain beef was instead containing horse meat, that is because, during the supply chain process, there was no recording of each production step and during the supply chain the information about the meat origin was lost, and at some point, someone assumed that was beef. Second, at CERN, during the Atlas project 100PB of experiments were accumulated. Since the reproducibility and accessibility of this massive amount of data was a primary concern, it was used as a strategy for preserving both the data and the procedures used to analyze them, relying on provenance.

Since provenance can be used for many different applications, the level of detail of the provenance information collected is different. Fig 7.12 shows four different levels of provenance where the levels range from a piece of provenance information very broad to a very specific, dependent on the application domain, and directly related to the level of instrumentation available to collect provenance of a given type. Remember that the transition from one type to another is a gradient rather than a sharp edge.

Provenance levels of details:

- *Provenance Meta-Data* is the most general type of provenance that encompasses any possible meta-data about a production process. Note that we distinguish provenance meta-data from other meta-data based on their intended applications. It provides users with the widest degree of freedom to model, store, or access the provenance of any type of end product or

production process and allows them to classify proprietary solutions for provenance management. Indeed, whereas general meta-data aims at assigning meaning to data, provenance information is descriptive of the data derivation process.

- *Information system provenance* is meta-data about processes within an information system that are involved in the dissemination of information. We define information system provenance as meta-data collected for an information-disseminating process that can be computed based on the input, the output, and the parameters of the process.

- *Workflow provenance* specializes the previously described information system provenance by further restricting the type of production processes to so-called workflows. We view a workflow as a directed graph where nodes represent arbitrary functions or modules in general with some input, output, and parameters and where edges model a predefined data flow or control flow between these modules.

- *Data provenance* is the "highest resolution" of provenance since it records the provenance in terms of individual data items.

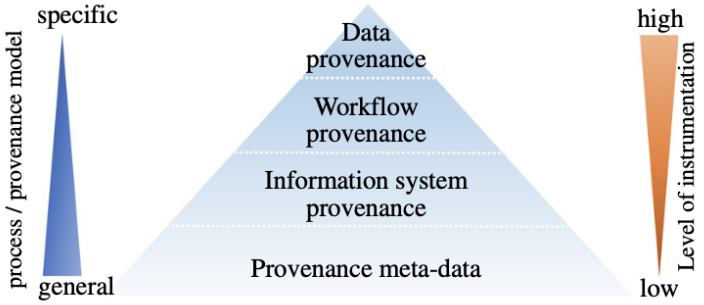


Figure 7.12: Provenance Level of details

Provenance has the potential, for example, to increase transparency and accountability within the machine learning community, mitigate unwanted societal biases in machine learning models, facilitate greater reproducibility of machine learning results, and help researchers and practitioners to select more appropriate datasets for their chosen tasks.

To represent the provenance of an object, we can mainly use two types of representation. The W3C consortium suggests the first, and it is called the Prov Data Model, while the latter is the data sheets format that is instead supported by Microsoft.

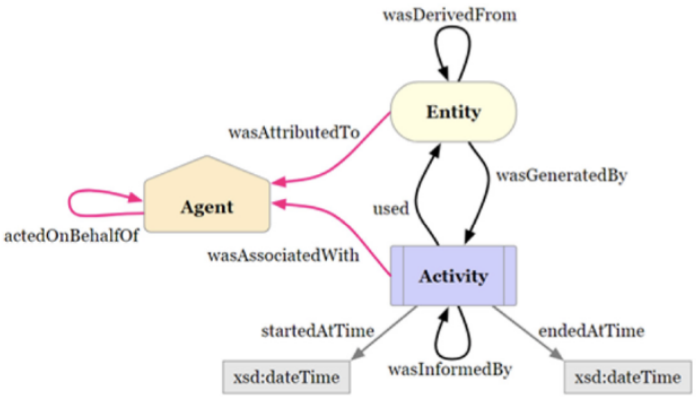


Figure 7.13: W3C Prov Data Model

W3C PROV Data Model

The W3C PROV data model is composed of the so-called Starting Point Terms and consists of three primary classes with unique and mandatory identifiers and nine properties to describe the relations between the classes. The three concepts are Entity, Activity, and Agent. An Entity is the main character of the provenance since it is the subject about which we would like to know its provenance. An Activity models actions performed on the entities. An Agent denotes persons or machines who bear some form of responsibility for an activity. Some relationships connect these concepts. The nine most important relationships as we can see from Fig. 7.13 are: used (an activity used some artifact), wasAssociatedWith (an agent participated in some activity), wasGeneratedBy (an activity generated an entity), wasDerivedFrom (an entity was derived from another entity), wasAttributedTo (an entity was attributed to an agent), actedOnBehalfOf (an agent acted on behalf of another agent) and wasInformedBy (an activity used an entity produced by another activity). W3C PROV data model has primarily been designed to describe retrospective provenance, which refers to a-posteriori descriptions of provenance traces of data resources, i.e., provenance as an extended log of all the steps executed to generate the data entity. However, the concept of provenance can also refer to tracing the genesis of workflows used for generating data, and moreover, even to the a priori description of such workflows, in which case it is called prospective provenance, which can be considered to be a form of workflow description language.

Data Sheets for dataset

Learning from the industry, every component that has been designed is accompanied by a datasheet containing different information such as operating

characteristics, test results, and recommended usage.

Datasheets for a dataset in a similar way want to mimic these habits and translate them into the computer science domain.

Basically, a datasheet is a text file that should answer specific questions about the dataset from different perspectives. The questions are divided into seven sections: motivation, composition, collection process, preprocessing/ cleaning/ labeling, uses, distribution, and maintenance. This grouping encourages dataset creators to reflect on the process of creating, distributing, and maintaining a dataset and even alter this process in response to their reflection. If a question does not apply to a dataset, it should be skipped.

- *Motivation* These questions want to encourage dataset creators to clearly articulate their reasons for creating the dataset and to promote transparency about funding interests.
- *Composition* Most of the questions in this section are intended to provide dataset consumers with the information they need to make informed decisions about using the dataset for their chosen tasks. Some of the questions are designed to elicit information about compliance with the EU's General Data Protection Regulation (GDPR) or comparable regulations in other jurisdictions.
- *Collection Process* With these questions dataset creators should make the final user aware of the collection process of the data to let them identify potential issues for their task. In addition, the questions in this section are designed to elicit information that may help researchers and practitioners to create alternative datasets with similar characteristics.
- *Preprocessing/cleaning/labeling* The questions in this section are intended to provide dataset consumers with the information they need to determine whether the "raw" data has been processed in ways that are compatible with their chosen tasks.
- *Uses* The questions in this section are intended to encourage dataset creators to reflect on the tasks for which the dataset should and should not be used. By explicitly highlighting these tasks, dataset creators can help dataset consumers make informed decisions, thereby avoiding potential risks or harms.
- *Distribution* These questions are about who will potentially use the dataset.
- *Maintenance* The questions in this section are intended to encourage dataset creators to plan for dataset maintenance and communicate this plan to dataset consumers.

7.4 Learning resources

- `https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/`

- https://gaia-x.eu/mediatech/videos/
- Provenance Data Model: https://www.w3.org/TR/prov-dm/
- https://s11.no/2020/prov/validating-and-visualising-prov/

Chapter 8

Datawarehouses

8.1 Datawarehouses

8.1.1 Data characteristics

A data warehouse is an OLAP system.

The analytical data used to support strategic and decision-making activities differ from operational data for several characteristics. Let us remind here the main ones discussed in the introduction:

- *Objective*: the data must be used to make decisions about the future development of the organization, but also to identify organizational problems or growth opportunities for the organization.
- *Users*: data users are high- and low-level managers who use the information to improve the organization's results.
- *Time horizon*: while operational data is primarily based on the present, analytical data uses historical data relating to current data to identify issues, trends, and periodicities.
- *Level of detail*: analytical data are usually aggregate data derived from operational data or other sources according to different levels of granularity.
- *Access*: unlike operational data, users of analytical data access information only in reading, and not in writing.

Given these characteristics, the classic relational model at the operational level is not satisfactory for building an analytical database. While you could run queries on operational data to aggregate them, the large volumes of data to be aggregated would be too heavy to be processed directly in the operational system in an acceptable time.

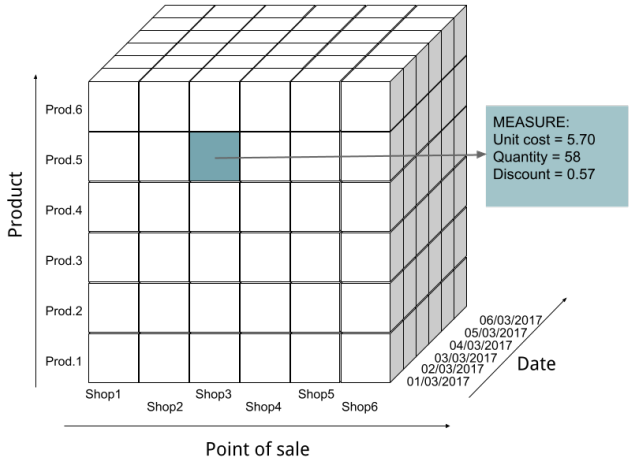


Figure 8.1: Three-dimensional hypercube representing the relevant aspects for the analysis of the sales of a supermarket chain

First of all, we have to identify in the data which are the analysis dimensions, i.e., the attributes that are of interest for analyzing the objects being considered. Not all the data contained in the operational databases will be considered, but only those which are needed for the analysis. It is also essential to define how the data can be aggregated (e.g., when considering time, we can aggregate data at the date, month, term, and year level, e.g., aggregating all the sales of a given product in one year).

The *multi-dimensional model* is used to represent the data of an OLAP system. In this model, the information is represented through the concept of *hypercube*, formed by a variable number of *dimensions* N, in which each dimension is an analysis dimension for the database. Each element is an object of the database, i.e., the recording of information, whose value is defined by the coordinates represented by the analysis dimensions. The building blocks of a multidimensional database are:

- *Fact*: the fact is the hypercube element obtained by specifying a value for each dimension.
- *Dimension*: coordinates of each element that correspond to analysis dimensions.
- *Measure*: quantitative value(s) of the elementary fact.

An example of a three-dimensional hypercube for the analysis of the sales of a supermarket chain is shown in Fig. 8.1, in which the analysis dimensions are the point of sale, time, products, while the fact consists of sales and measures are the quantity sold, the unit cost, and the discount applied. Each dimension represents one of the coordinates of the fact and has a domain of discrete values

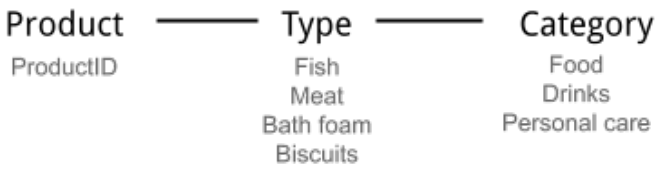


Figure 8.2: Example of hierarchy for the product dimension. The level of aggregation increases from right to left (finer to coarser).

that make it possible to define the set of facts of interest. If the values of a dimension are not discrete, they must be made so by applying, for example, thresholds to define the ranges of values considered. Dimensions can be numerous and organized into hierarchies, where the level of granularity varies from level to level. Dimension hierarchies are organized in such a way that there is a functional dependence between a level with higher granularity and a level with lower granularity, such that a relationship of the type $1...N$ exists between them. In the example of Figure 8.2, the products are divided into several categories. Each category includes one or more types of products, and each type includes multiple specific products. Along a hierarchical line, the relationships determine different aggregations for the considered dimension. Returning therefore to the hypercube of Figure 8.1, the product dimension can be manipulated by changing the level of granularity, for example by considering not single products, but grouping them by category (or by type). In this case, the content of the fact will be the aggregation of information for all products belonging to the considered category. Similar operations on other dimensions, their granularity or their values allow you to select the level of detail of interest for the analysis (e.g., aggregate sales in time or in space to analyze sales looking at locations of shops).

8.1.2 Data warehouse

The data warehouse is a database that collects synthetic, integrated and structured data of interest to an organization. In a broader sense, data warehousing can instead be defined as the set of all operations aimed at the population of the database, its maintenance, and all the query techniques that allow information to be extracted for analysis purposes. The data warehouse is an OLAP (On Line Analytical Processing) type system. The OLAP paradigm is characterized by few transactions and complex queries that require aggregation of large amounts of data. OLAP systems have response time as a measure of efficiency and effectiveness. An OLAP system stores data in aggregate format, stores historical data, archived according to multi-dimensional schemes. Queries often access large amounts of data to answer complex questions such as, for example, "what was the company's net profit in a certain geographic area in the past year?".

OLAP systems are used for decision support-oriented data processing, so

they are adequate for functionality located at the planning and strategic level of Anthony's pyramid. They allow to carry out complex and ad-hoc operations, in which every single operation can involve a lot of aggregate, historical, and generally even non-current data, and for which the ACID properties of a DBMS are not relevant because the operations are read-only.

An OLAP system, on the other hand, has the FASMI (Fast, Analytical, Shared, Multidimensional, Informational) properties. It must therefore have a multidimensional view of the data (*multidimensional*) which must contain all the information of interest (*informational*) on which it allows to perform complex analyses (*analytical*) to more users but with different data access permissions (*shared*) responding to their requests in a short time (*fast*).

In addition, the data warehouse must have the following characteristics:

- **Entity (or object) oriented**: considers the main analysis entities. For example, sales, orders, and products can be entities of analysis.
- **Integrated**: the data considered are taken both from internal sources and from sources external to the company. Starting from non-homogeneous data, they must be made consistent and integrated in order to have a complete and clear view of the entities analyzed.
- **Time-varying**: The data warehouse stores historical data. To analyze the historicity of events, all data are associated with a time label that identifies the reference period.
- **Persistent**: Once placed in a data warehouse, data is usually not changed. The data is stored in read-only mode.

8.1.3 Data warehouse architecture

Data warehousing is the set of activities that lead to the definition, construction, and maintenance of information in the data warehouse, which can be defined as the database that collects all the data of interest to the company, synthesizing, integrating, and structuring them. Within the data warehouse (DW) it is possible to identify different databases organized hierarchically. At the first level, we have the *sources*, i.e., the sources from which the data that will populate the data warehouse must be extracted. These sources certainly include the operational database of the organization, but also other databases or sources external to the organization itself that can be structured according to various forms (databases, reports, big data, etc.). All these sources are subjected to a set of operations called *ETL* (Extraction, Transformation, and Loading) which allow you to extract the data of interest from the sources, transform them according to the desired multidimensional structure, and load them into the data warehouse. This population phase can make use of an intermediate database between the sources and the actual data warehouse, called *staging area*. The data warehouse, on the other hand, is the multidimensional database containing all the information of interest accessible through the enhancement of the

analysis dimensions. Depending on the presence or absence of the staging area, the DW architecture can be two (without staging area) or three (with staging area) levels. Downstream of the data warehouse, there may be another type of database called *data mart*. Data marts are small thematic data warehouses containing an extract of the information contained in the complete data warehouse. The latter can have very large dimensions that are hardly of interest to all users of the system. Using data marts, you can define data warehouse views that contain a subset of the information. This subset can be a reduction of the analysis dimensions (only those of interest to the category of users to which the data mart is addressed are considered), of the level of granularity of the dimensions, or of the selected time horizon. An architecture that uses only DW is a centralized architecture, in which all data is contained in a single centralized multidimensional database. On the contrary, the use of data marts allows the use of a distributed architecture in which the different thematic databases can be positioned on different nodes.

To give an example, we can refer to a supermarket chain that uses data warehousing to improve its company. In the example, the sources of information can be various. First of all, there are the operational databases, in this specific case the database containing the sales data and the inventory database. Furthermore, information relating to the advertising campaigns conducted by the chain may also be of interest to the analyses. Other sources that can influence decisions may instead come from outside such as weather data (the sale of some products may also be affected by weather conditions), and data from specialized sales analysis agencies, such as Nielsen. This information is transformed and restructured using ETL transformations to be stored in the corporate data warehouse. From the interaction with the data warehouse, decisions can be made that are influenced by all the sources analyzed, such as supply planning. Other sectors, such as customer management or logistics planning, instead use other subsets of the data contained in the data warehouse and for this reason, we can think of the creation of thematic data marts that can be queried through an OLAP engine and from which it is possible to extract reports in an automated way.

ETL

Particular attention should be paid to the tools supporting the ETL process. These, as mentioned previously, allow you to extract, transform, and load data from the data warehouse. As for *extraction*, it defines which data is to be extracted and how it is to be handled (for example, whether it is to be aggregated at source or extracted at the highest level of detail). Furthermore, the extraction can be: (a) *Static* if all the data present in the sources are considered or (b) *Incremental* if only the data produced or modified by the sources in the range are considered time elapsed since the last update of the data warehouse.

Following the extraction, the data need some *transformations* as deriving from heterogeneous sources they can be represented in different formats and consequently could not be easily integrated. The main operations that are

performed are:

- *Format standardization*: data must be made homogeneous with standardization operations. These operations refer to (i) Conjunction or splitting of fields: information present in several fields is traced back to a single field or the information present in a field is reported in separate fields; (ii) Standardization of classification codes: different conventions to indicate the same criterion are aligned to the same code; (iii) Standardization of the data format: data represented with different formats are made homogeneous.

- *Data cleaning* (data cleaning): The data extracted from the sources many times contain errors that must be corrected before being inserted into the data warehouse. This is usually done first in the transformation phase, but in some cases, it can also be done afterward. The main problems related to the quality of the data that are solved in this phase are the lack of data, the presence of not admissible values, and the inconsistency between values present in different fields, but which are linked together by particular rules or which have the same meaning.

- *Reconciliation*: the integration of different data must be preceded by this operation which aims to relate the data to the same "object".

- *Search and elimination of duplicates*: in case of duplicate information, this operation allows to identify them and reduce them to a single instance.

The *loading* phase takes care of transferring the data into the data warehouse following the multidimensional model introduced in Section 8.1.1.

Metadata

The functionalities performed by the ETL tools are supported and documented by metadata (data related to data). Metadata is an important part of a data warehouse architecture and is usually collected in a repository that includes:

- *Structure of the DW*: the metadata describes the structure of the DW (diagrams, views, dimensions, hierarchies, the decomposition into data mart, and relative localization).

- *Operational metadata*: refer to the history of the data and document the source of origin and the transformations applied.

- *Metadata to map operational data to data loaded into the DW*: they provide information on the sources and their content, the transformation and updating rules of the DW.

- *Usage statistics*: The metadata also describes how and how often the DW is used.

8.1.4 Conceptual model of the data warehouse

As previously mentioned, data warehouses are based on the multidimensional model of data in which the facts are represented, which are the elements of interest of the analysis through a set of measures, defined by the dimensions which are the coordinates of the fact.

There are several conceptual models to represent multidimensional systems. The most used is the Dimensional Fact Model (DFM). In the DFM, the fact is represented as a rectangle containing the corresponding measures. Instead, dimensions are represented as labeled circles linked to the fact itself. Dimensions can be simple attributes of fact or hierarchies, represented as trees (or graphs), rooted in the basic dimensions. Some attributes of the hierarchies can be optional and this is indicated in the DFM by a bar on the line corresponding to the attribute.

An example of DFM representing sales for the supermarket chain mentioned above is shown in Figure 8.3. The fact in this case is represented by the sale which measures the quantity sold, the unit cost, and any corresponding discount. The fact is determined by four dimensions: the date on which the sale was made, the place where the fact occurred, the product being sold, and the customer. A hierarchy is associated with each of these dimensions, through which it is possible to carry out more or less detailed analyses of the facts recorded in the data warehouse. For example, it is possible to analyze the total quantity of sales relating to a specific product for the entire analysis period for the points of sale relating to the city of Milan grouped according to the gender of the customers. This information results from the aggregation along the various hierarchies of the considered dimensions. In the modeled DFM, the age attribute for the customer is optional.

The DFM shown as an example corresponds to the sales hypercube for the supermarket chain (with the addition of customer information here). The data warehouse, however, will not be composed of a single hypercube, but of many hypercubes, each capable of modeling the dimensions relating to a fact of interest for analysis. For example, the data warehouse of a company interested in analyzing sales, customer support efficiency, and optimizing supplier interaction will have three different hypercubes in its data warehouse, one for each analysis of interest: sales, customer service, and suppliers. A DFM will then be modeled for each of these hypercubes, with the definition of the fact, of the measures, of the dimensions, and of their hierarchies of interest for the single facts under analysis.

8.1.5 Logical models of the data warehouse

Once the conceptual model of the multidimensional database has been defined, it must be implemented by translating the conceptual model into a logical model. It is therefore necessary to choose which DBMS to use to manage the multidimensional information.

Several technological solutions are possible, in the following we discuss the

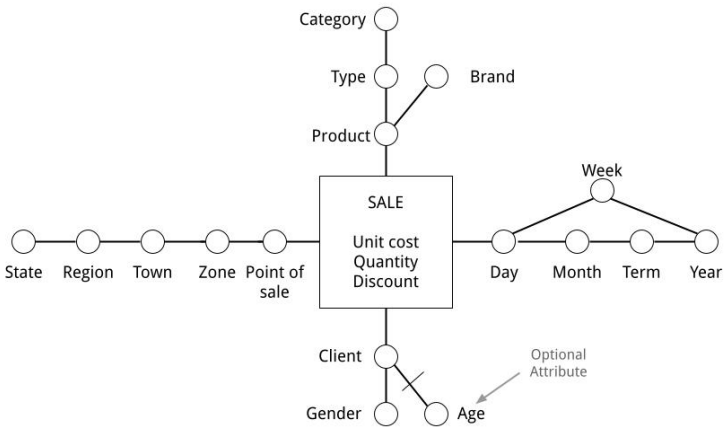


Figure 8.3: Dimensional Fact Model for supermarket sales

representation based on the relational model (ROLAP). The DFM shown as an example corresponds to the sales hypercube for the supermarket chain. The data warehouse, however, will not be composed of a single hypercube, but of many hypercubes, each capable of modeling the dimensions relating to a fact of interest for analysis. For example, the data warehouse of a company interested in analyzing sales, customer support efficiency, and optimizing supplier interaction will have three different hypercubes in its data warehouse, one for each analysis of interest: sales, customer service, and suppliers. A DFM will then be modeled for each of these hypercubes, with the definition of the fact, of the measures, of the dimensions, and their hierarchies of interest for the single facts under analysis.

Multidimensional schemes on relational databases To map a multidimensional database into a relational schema it is necessary to identify the tables that will make up this schema. One approach to this in the literature is known as the *star schema*. In the star schema, two types of tables are used: the fact table and the dimension tables. The first contains all the attributes corresponding to the measures of the fact and each row of this table will therefore correspond to a specific fact. The dimension tables, on the other hand, contain, for each dimension associated with the fact in the multidimensional schema, all the attributes relating to the hierarchy corresponding to the dimension. A query will then result in a JOIN between all tables involved based on the fact of interest. In this way, however, the concept of a hierarchy of dimensions is lost, as it is flattened within a single table. An example is shown in Figure 8.4 which shows the star scheme relating to the multidimensional model whose conceptual model is shown in Figure 8.3.

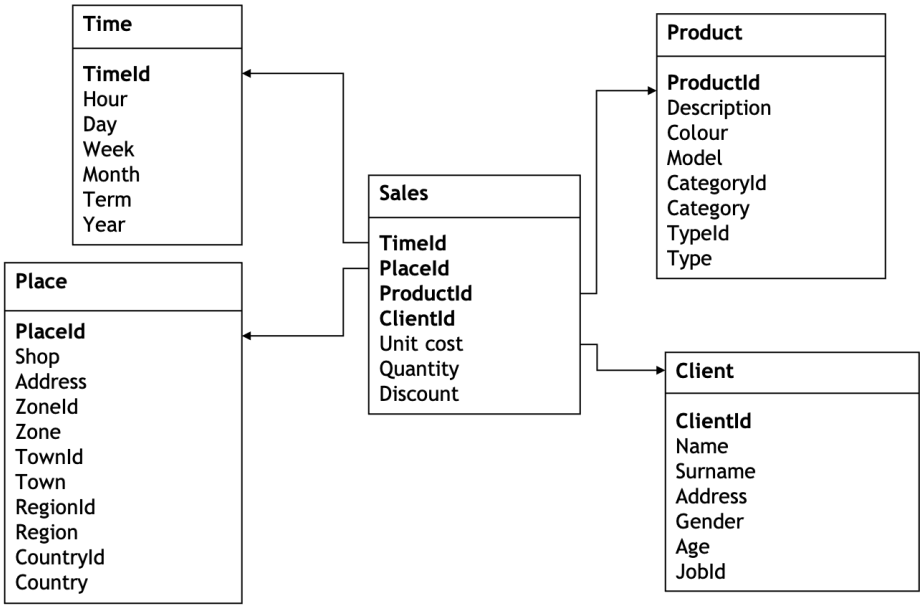


Figure 8.4: Star schema to represent the sales fact of a supermarket chain

8.1.6 Data warehouse operations

In addition to a multidimensional data model, data warehousing also includes a set of techniques that can be used to analyze data. These techniques must allow even inexperienced users to interact with the hypercube of facts to obtain more or less detailed information based on needs, focusing on the dimensions of interest. The interaction usually starts from a hypothesis formulated by a user which leads to the extraction of the subset of the data of interest that can be deepened with subsequent interactions. Each interaction consists of applying an OLAP operator. The main OLAP operators are: drill down, roll up, slice and dice.

The **drill-down** is the operation that allows you to obtain more detailed data by going down a hierarchy of a dimension and then passing from a higher to a finer level of aggregation. Considering the DFM in Figure 8.3, a drill-down operation could switch from an analysis by terms to an analysis by a single day of sales (see Figure 8.5). The opposite operation to drill down is that of **roll-up**, where the level of detail is reduced to a coarser level of granularity along an analysis dimension. An example could be to go from a daily sales analysis to sales aggregated in terms in which the values per day are aggregated together.

The **slice** operator allows you to focus the analysis on a portion of the data by setting the value for one of the analysis dimensions. For example, it is possible to focus on sales relating to a single store, while maintaining all the details relating to the other dimensions of analysis. The result of this operation is a slice of the original hypercube, identified by the fixed dimension (see Fig. 8.6).

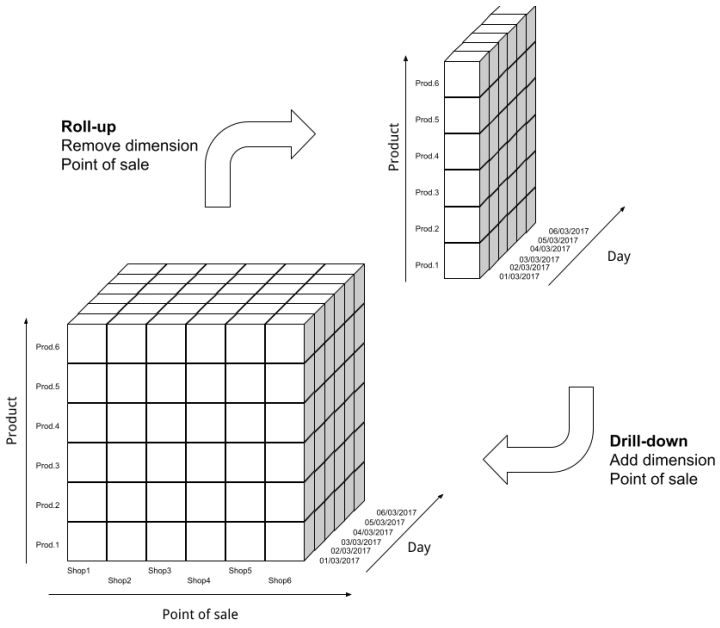


Figure 8.5: Example of roll-up and drill-down with aggregation on the dimension *Date*

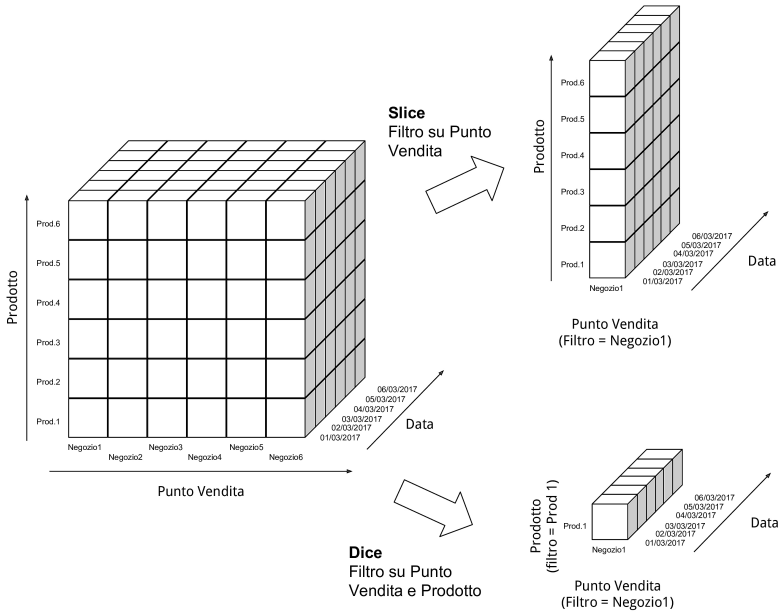


Figure 8.6: Example of slice and dice

Note that, unlike the roll-up presented above, the slice selects a single store and does not aggregate concerning the store. Another reduction operation is that of **dice** in which a set of coordinates at any hierarchical level for any desired dimension are identified. The result in this case is always a hypercube but with a reduced set of data. For example, you can analyze the sales data for a product (e.g., *product1*) in a particular store (for example *shop1*).

8.1.7 Data warehouse life cycle

As mentioned in Section 8.1.4, the data warehouse of an organization will be composed of many hypercubes, each containing information relevant to a fact of interest for the management of the organization.

The construction of the DW therefore follows an iterative approach. We start by identifying the most interesting fact and then the first hypercube is modeled and populated. Subsequently, one at a time, other hypercubes of interest are added and in turn populated. This approach has a double advantage: on the one hand, it makes it possible to identify the facts of interest in subsequent moments, responding to new needs of the organization; on the other hand, it limits the initial investment by developing the multidimensional database one piece at a time.

The DW population phase is a complex task that requires the execution of several operations. As already said, to populate the DW we use the ETL tools (Extraction, Transformation, and Loading) (Section 8.1.3), where the last step is *loading* the data into the DW.

During the insertion, it is also important to consider the problems related to the updating of the data which may concern facts or dimensions. Updating a fact means entering a non-existent value for a combination of dimensions, or modifying an existing value by updating it. Updating a dimension can instead lead to an inconsistency. For example, if the address dimension for a customer changes over time, one of the following decisions will need to be made: (1) update the address value for pre-existing facts as well so that a customer is always associated with the same address; (2) keep the previous value valid without performing any updates; (3) adopt a hybrid solution in which traces are kept of the various addresses relating to the customer related to the relative time intervals. The last solution is the most complex one to manage.

The life cycle of a data warehouse also highlights its limitations. It is important to underline that the ETL process can take some time to execute. It is not necessarily an automatic activity. Especially in the transformation phase, where it is necessary to resolve some discrepancies between data from different sources, human intervention may also be necessary. Therefore, not only is the creation of a data warehouse a costly operation but also its maintenance and continuous updating of data requires a fair amount of effort.

This complexity produces, as an effect, an increase in the distance between the updating of data in source databases - which happens very frequently - and the moment in which these updates are visible in a data warehouse. To reduce this distance, in recent years, alternative approaches to data warehouses have

been proposed that often exploit algorithms and techniques typical of big data which, while not requiring such a detailed design phase as in the case of the data warehouse, can produce analysis results in real-time, as it will be discussed in next section and in following chapters.

Chapter 10

Introduction to security

Mariagrazia Fugini, Barbara Pernici

10.1 Introduction to security

This chapter is organized in three parts. In the first part the basic concepts are introduced, such as threats, attacks, access to systems and authentication, the role of cryptography, digital signature, identification.

In the second part privacy laws and access to data are discussed.

In the third part (cybersecurity), trustless environments and cybersecurity are discussed, and transactions and blockchain illustrated.

10.1.1 Basic concepts

Methods and techniques for security are more and more relevant. Around 90% of intrusions into computers and networks remain undiscovered. According to a research performed by DoD, attacks are successful in 80% of cases; in only 6% of cases, intrusion has been detected. The main problem is that systems are inherently weak. It is hence necessary to set up *Security Policies*, *Technical Mechanisms*, and *Organizational Measures* to face security issues globally.

A useful concept to design, implement and manage security is the concept of *security domain* for an organization (Fig. 10.1).

One the one side, there is the need to share and cooperate in networked systems belonging to various organizations. On the other side, sharing and diversity of applications, information and resources, preserving existing (legacy) systems and databases, with the requirements of autonomy and interoperability introduce various security problems, that are illustrated in what follows.

In an organization, several *vulnerabilities* are present. These are present in the three different areas that can be defined in general: the internal network, where *high security* is required, external areas connected via Internet, with *low security*, and external but protected areas, such as extranets, with a *medium security* level (Fig. 10.2).

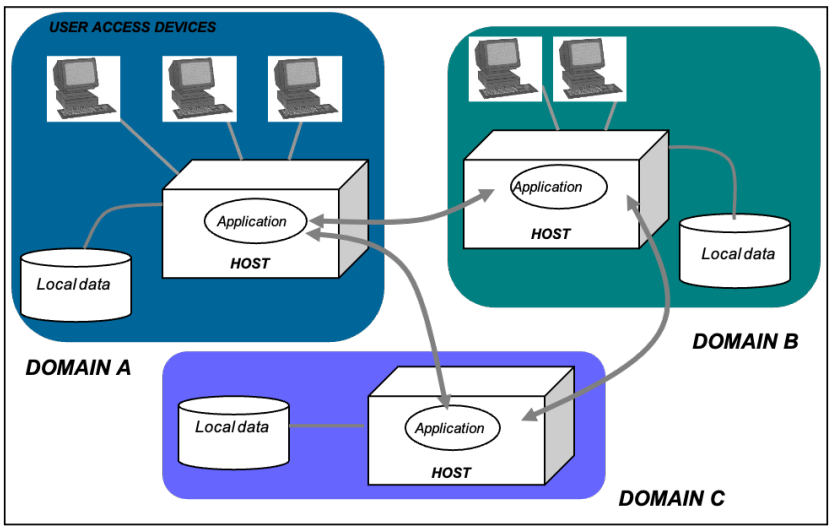


Figure 10.1: The concept of Domain

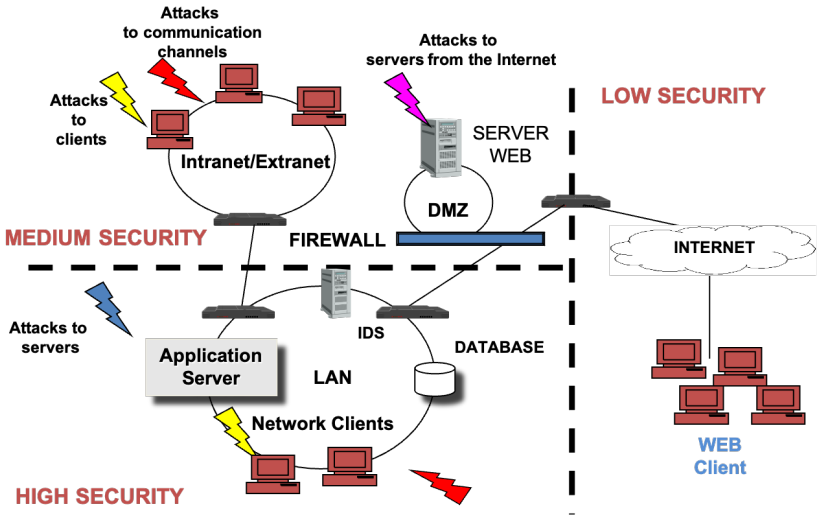


Figure 10.2: Vulnerabilities in complex systems - One company perspective

10.1.2 Threats and attacks

Threats

Threats can be classified in three types:

- *Physical*: e.g. thefts, intentional damages, accidental events (air conditioning malfunctioning), natural disasters
- *Logical*: e.g. Data stealing (credit card numbers), hidden access points to the system (illegal usage of resources)
- *Accidental*: e.g. misconfigurations, malfunctioning of programs, data entry errors.

Attacks

In the context of information systems, security *attacks* are any action, carried out by individuals or organizations, which affects databases, infrastructures, communications networks and / or personal electronic devices through malicious acts aimed at the theft, alteration or destruction of specific elements (defined as attack targets), violating the rules of access to systems. This becomes particularly relevant in systems defined as critical (for example, government, health, military, etc.).

Several types of attacks can be classified. The goal of an organization is to be able both to prevent attacks and to detect attacks and react to them. Causes of attacks can be weaknesses in technical design or implementation, inadequate or not applied management procedures, misbehavior of users, and so on.

A first classification distinguishes between attacks aiming at reading data without authorization and attacks modifying data. In the following examples we show such attacks in a communication channel, with messages M sent by a sender (referred to A or Alice in the literature) to a recipient (referred to B or Bob in the literature).

Sniffing: packets are captured by unintended receiver (reading data, e.g., passwords) (Fig. 10.3). This attack is difficult to detect until the intruder makes any unauthorized use of the data. The solution is making the message unreadable by a third party, using cryptography.

Other types of attacks modify the message:

- *Address spoofing*: packets are generated with false sender information. Solution: strong authentication mechanisms.
- *Data spoofing*: false or modified data are created.
- *Hijacking*: also called Man-in-the-middle, is a form of data spoofing, creating an alternative channel (Fig. 10.4).

Attacks can be perpetrated by different actors:

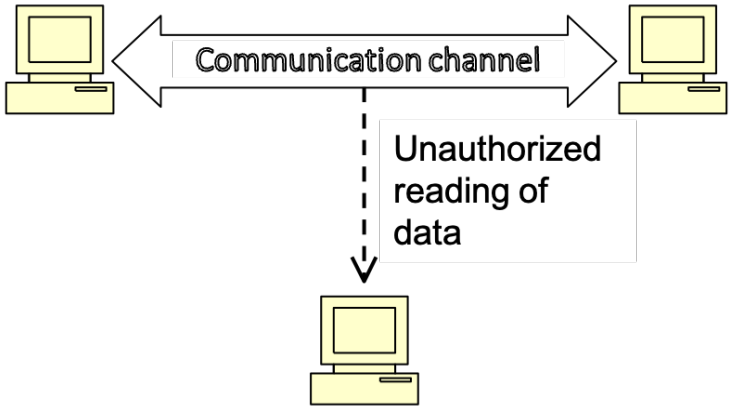


Figure 10.3: Sniffing attack

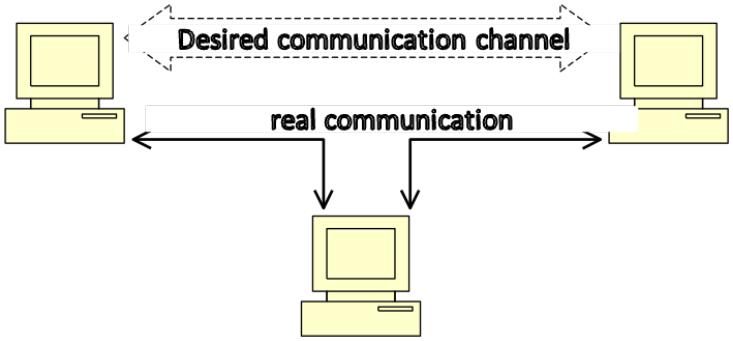


Figure 10.4: Hijacking

- Attacks from third parties come from various points of access to the network.
- Network manager: cancelling/altering network traffic.
- Non-authorized personnel
- ... also legitimate users.

General principles in security management

Security management has some common general principles:

- Cross-layer: when considering layered system architectures, such as for instance the one proposed for Big Data by BDVA presented in the chapter discussing architectures, security is considered to be a cross-layer issue. It is not sufficient to consider it separately in one or more layers, but it should be handled in coordination at all levels.
- Monitoring and control: as security is based both on prevention and detection, it is necessary to continuously monitor the system and apply control rules and countermeasures.
- Managed services: all services provided in the system have to be properly managed, including both IT services but also other services including electricity and network connections, as well as services based on human providers.
- Often security considered a quality property: Quality of Service (QoS) often includes security clauses and penalties, as it is essential for the functioning of the system.

10.1.3 Security properties

In defining security properties, it is essential to introduce first the definition of the two main elements on which security properties are based.

Information: any sequence of bits useful for the functioning of the system (data, executable code, configuration of applications)

Agents: Any entity able to operate autonomously on the system (users, applications or processes). *Authentication* requires that agents are identified when accessing the system.

A formal definition of information systems security is that provided by ISO [11] which considers it as the set of measures to protect the requirements you want the system to meet. These requirements are defined according to the *security properties* described below:

- *Integrity:* the system must prevent the direct or indirect alteration of information, both by users and unauthorized processes both as a result of accidental events.
- *Authenticity:* the recipient of the information must be able to verify the identity of the sender and the information must be intact. The author of the information cannot deny the authorship of the information (*non-repudiation*).
- *Confidentiality:* information read only by authorized agents. The term *privacy* is referring to confidentiality of personal information.
- *Service availability:* system can provide the services for which it has been designed when requested by the users.

10.1.4 Access to systems: Authentication

Before each security check, agents must *authenticate* before allowing them to open a communication channel or access information located on the Intranet or in secure areas. The authentication mechanisms allow the system to recognize an agent in a certain way, requesting credentials that lead him to a more advanced step in accessing the system, compared to a simple registration (used for example, to browse information). With authentication, filtering is possible in the data access modes of applications and from the DBMS or other components of the information system.

An agent can be identified by exploiting one or more of the following factors:

- SOMETHING YOU KNOW - SYK: known information (e.g., password).
- SOMETHING YOU HAVE - SYH: a possessed item (e.g., a magnetic card).
- SOMETHING YOU ARE - SYA: a physical characteristic - in the case of people (e.g. fingerprint, retina, DNA).

Simple authentication is based on userid and password.
Robust authentication is based on several possible mechanisms:

- One-Time Password (OTP) mechanisms
- challenge-response systems, in which the password is never transmitted but only used to make a calculation which indirectly demonstrates knowledge: answer = F (challenge, password), or through the use of personal devices (tokens) as occurs in smart cards
- Two-Factor Authentication (2FA) and MFA (Multi-Factor Authentication)
- on biometric characteristics

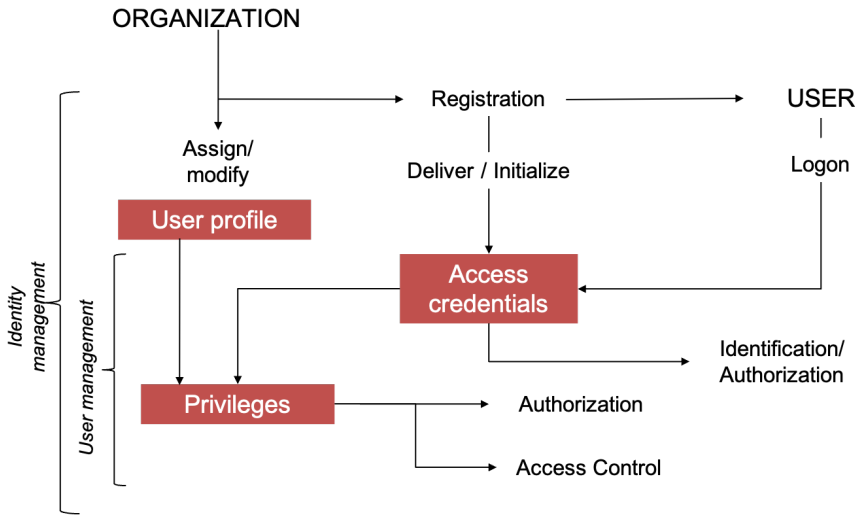


Figure 10.5: User management

Authentication can also be based on the use of public key certificates (see below).

While authentication allows you to verify the license of an identifier (credentials), by associating an agent with a specific *identity*, *authorization* is the phase in which it is verified whether the agent is authorized to perform the required function, such as, for example, operating system functions (for example, file access), application functions (business process, task) and programs.

In particular, *access control* is the set of procedures that check whether the agent is authorized to perform elementary functions (read, write, execute). It can be done by the operating system, the DBMS or by specific security programs / services, thus limiting the damage that can be caused by illegitimate users and / or malicious programs because the security services are designed specifically for the resources to be protected.

Figure 10.5 offers an overview of secure user management through the various mechanisms illustrated so far and in the rest of this section.

Within an organization, users have credentials to access the system, that give access with the methods discussed above.

Upon registration, users are also associated with a profile (e.g., in a university system, profiles can be student and professor). For each profile, some privileges are assigned, defining to which data and which service can each user access, and with which operations (CRUD - Create, Read, Update, Delete - operations for data, RWX - Read Write Execute - operations for applications) and with which rules (e.g., both students and professors can read grades. However, in the case of students, they can see their own grades, while in the case of professors, they see the grades of the students the classes they teach).

10.1.5 The role of cryptography

Among the fundamental techniques, cryptography certainly plays a role of primary importance. Thanks to cryptography it is in fact possible not only to protect the confidentiality of transmissions (confidentiality properties), but also to provide solutions for authenticity and integrity. Therefore, the basic principles governing the algorithms usually used are illustrated below.

Encrypting means encoding information, transforming the original message into an encrypted message. The encryption mechanisms consist of an algorithm (*cryptographic function*) and one (or more) *keys*. Only the secrecy of the key (or one of the keys) can guarantee the confidentiality and authenticity of the messages. Furthermore, the key must be chosen from a very large number of combinations (key space) and changed often (for example, session key). Cryptographic security depends on the breadth of the range of possible values that a key can take: the more possible values, the more difficult it becomes for unauthorized persons to recover the key and decrypt a message.

There are two classes of algorithms:

- *Symmetric key (or secret key) algorithms,* to which substitution algorithms, transposition algorithms and algorithms deriving from their combination belong. For these algorithms there is only one key, called *secret* or symmetric, which must be shared securely between the sender and the recipient. Examples for this class of algorithms are the DES (Data Encryption Standard), actually used in the 3-DES (Triple-DES) version, and AES (Advanced Encryption Standard) which became the standard algorithm in 2000.

- *Asymmetric key (or public key) algorithms,* which are based on mathematically complex problems and on the fact that each subject has a pair of keys (*private key* and *public key*). For this class of algorithms, the most used methods are RSA and Diffie-Hellman.

Symmetric cryptography

In Fig. 10.6 the scheme of operation of the symmetric cryptography is shown, where K indicates the symmetric key (*secret key*) shared between sender and recipient. In case of sniffing attacks on the insecure channel, the attackers, failing to capture the plaintext M, could only collect a large number of messages on which to do cryptanalysis with the intent of deriving the key K. To make cryptanalysis useless and to guarantee the security of communication, the K key must change often (possibly at each session). The problem is that A and B have to find an alternative secure channel to the one used for the transmission to exchange the secret K key frequently.

The two main techniques behind symmetric cryptography are substitution and transposition (or permutation).

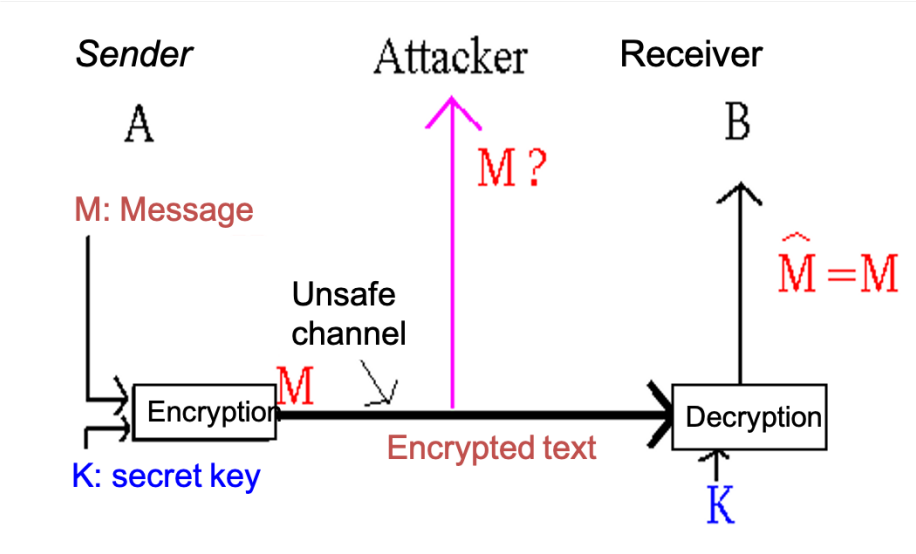


Figure 10.6: Symmetric cryptography

Substitution. Figure 10.7 shows two mechanisms of fixed *substitution*, with numeric key and with monoalphabetic substitution. In general, mappings with longer texts are necessary to make it more difficult to reconstruct the original text. A simple symmetric key algorithm based on the fixed substitution mechanism is the Caesar cipher. The encryption algorithm expects the message to be encrypted by replacing each letter of the message with the letter of the alphabet found k positions below, where k is the key to be shared between the sender and recipient. For example, if sender A wants to send a message M = "Fugini" and $k = 3$, the message M' will be equal to "IALNQN" (considering the Italian alphabet).

Transposition. In *transposition*, the key is used to indicate how to permute the letters contained in the plaintext during encryption. In the example shown in Figure 10.7, the key indicates in how many columns the plaintext is transcribed, which are then sorted according to the lexicographical order of the letters of the key. So the first letters of the cyphertext correspond to the column that contains the letter A of the key: "PROVATI" and so on, following the order indicated for the columns by the key.

Substitutions and transpositions are the basic operations of symmetric cryptography algorithms, as shown in Figure 10.8, in which the encrypted message is obtained from the plaintext by a series of substitutions - blocks S-box - and transpositions - P-box blocks (Permutation box).

The standard algorithm is AES where the message is encrypted using a key that can be 128, 192 or 256 bits long. For encryption, the message is split into 128-bit blocks and each block is encrypted through several steps which include

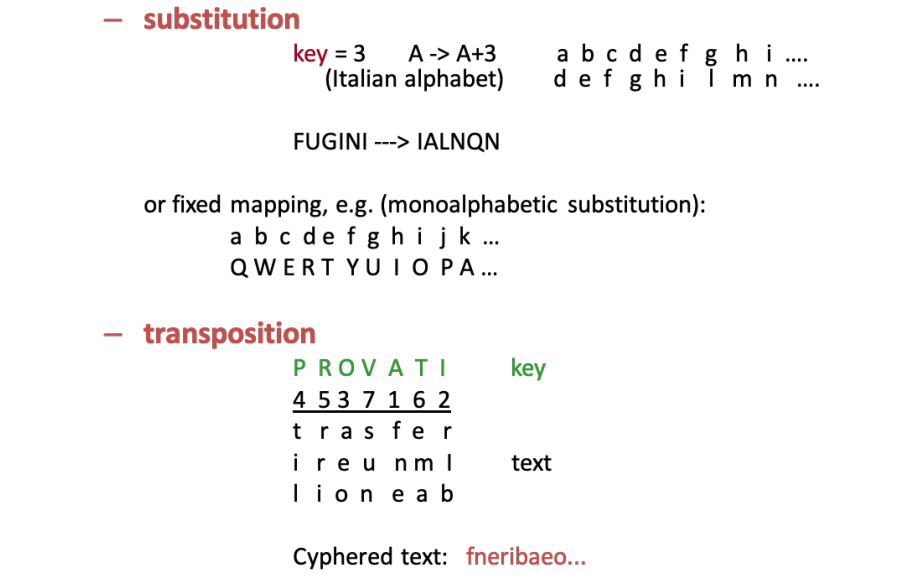


Figure 10.7: Symmetric Cryptography: substitution and transposition

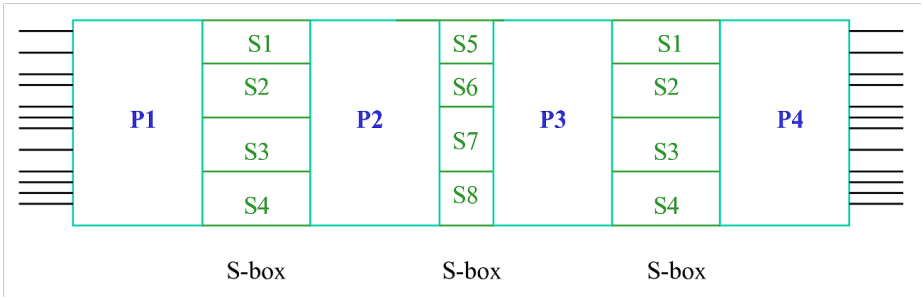


Figure 10.8: Combination of S-box and P-box blocks

xor combinations of the message bits with key bits, substitutions, transpositions and bit combinations in intermediate steps.

The critical points of symmetric cryptography are related to key management: (i) the key must be changed often to prevent it from being discovered, (ii) the key must be generated and distributed with maximum security. The main advantage of symmetric cryptography is the speed with which the encoding and decoding operations are performed, as it can also be implemented in hardware with dedicated processors.

Asymmetric cryptography

Asymmetric (or public key) cryptography requires that each agent has a related key pair: a *public key* K_{pu} and a corresponding *private key* K_{pr}. The correlation between the two keys relates to the principle of an asymmetric encryption algorithm: anything that is encrypted with one of the two keys can be decrypted with the other key of the pair. So, assuming A's holding key pair is $\{K^A_{pr}, K^A_{pu}\}$ everything encrypted with K^A_{pr} can only be decrypted with K^A_{pu} and vice versa. An important property of key pairs is that it must in no way be possible to deduce one of the two keys from knowledge of the other.

The public key can be distributed, while the private key must remain secret and the agent who owns it must keep it properly. As it is not possible, due to the property mentioned above, that given the public key it can be traced back to the private one, although the first is available to anyone on public repositories, the security of the second is however not undermined. This avoids the need to transmit secret keys between agents on the network, who only have to share their public keys, either directly or through a Public Key Distribution Service.

With asymmetric cryptography, we can use the two key pairs, $\{K^A_{pr}, K^A_{pu}\}$ and $\{K^B_{pr}, K^B_{pu}\}$, associated to A and B respectively, to ensure confidentiality and authenticity.

In particular, a first way to use symmetric key algorithms is shown in Figure 10.9. In this case, sender A encrypts the message with B's public key and, therefore, the message can only be decrypted with B's private key. This guarantees the *confidentiality* property as B will be the only one able to decrypt the message (being the only one in possession of K^B_{pr}) and consequently the sender is guaranteed that the message will not be read by unauthorized agents. The weakness of this mechanism is that everyone can get hold of B's public key and consequently pretend to be A within the communication, with problems of Man-in-the-middle attacks.

Sender A could also encrypt the message with his private key. In this case, recipient B must decrypt the message with A's public key (see Figure 10.10). In this way the exchange is *authenticated*, since sender A certifies his identity by encrypting with his own private key.

However, the message written by A is easily readable even by third parties in possession of A's public key. To guarantee both the authenticity property and the confidentiality property, the two approaches described above must be combined with a double encryption phase.

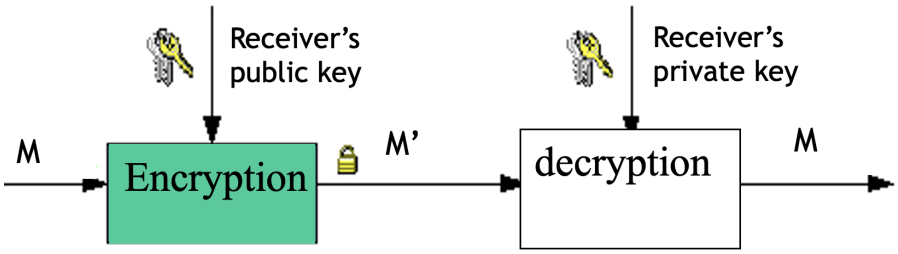


Figure 10.9: Asymmetric cryptography (confidentiality)

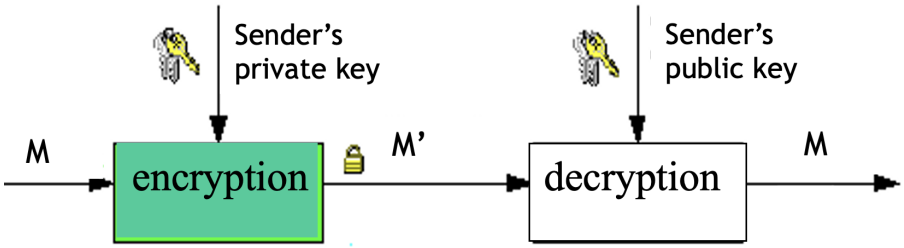


Figure 10.10: Asymmetric cryptography (authenticity)

The disadvantage of the asymmetric key algorithms in general is the computational complexity. In fact, the need to calculate, for instance, the exponential of non-small numbers, requires a certain computational capacity for very long messages. For this reason, asymmetric encryption is often used to encrypt the keys that must be exchanged between agents in private key algorithms, and then symmetric encryption is used for the exchange of messages or texts.

10.1.6 Digital signature

The goal of digital signatures is to guarantee authenticity.

Three elements are needed:

- Asymmetric keys (public + private)
- Digital certificates (to link public keys to agents)
- Hash functions (to compress messages in single blocks)

Hash functions

The symmetric and asymmetric encryption algorithms guarantee the properties of confidentiality and authenticity. However they work on the single block level,

Field	Example
version	1
serial number	2430
signature algorithm	RSA with MD5, 1024
issuer	C=IT, O=PoliMi
subject	C=IT, O=PoliMi, CN=MariagraziaFugini
validity	1/1/2021 - 31/12/2023
subjectpubickeyinfo	RSA, 1024, xxxx…xxx
digital signature	yy…..yyy

Figure 10.11: Digital certificate example

while messages and documents are usually composed of several blocks. To guarantee integrity we need a mechanism to provide a summary of a message in a single block.

To be able to provide also a guarantee in terms of integrity, we make use of the *hash functions* . A hash function is a function that transforms an arbitrary length M message into a fixed length output called *fingerprint*, *hash*, or *digest* of the original message. The *hash* algorithms are public and well known. The hash function is used in this context as it is characterized by the following properties:

- Consistency: identical messages must be associated with the same digest.
- Uniqueness: the probability that two different messages (even of a single bit) are associated with the same digest must be almost zero.
- Non-invertible: the function must not be invertible. It must be impossible to trace the message starting from the digest.

The hash functions thus generate the fingerprints of the message, which can constitute proof of the integrity of the message itself, that is, of the fact that it has not been manipulated by unauthorized agents during communication.

The hash function by itself does not guarantee integrity, though, without using cryptographic mechanisms, as it is a text that can be attacked.

Public Key Certificates

Public Key Certificates have the goal to guarantee that a given Public Key is associated to a given Subject.

A simplified example of a certificate is shown in Figure 10.11, where the various fields also indicate the encryption algorithm used (RSA), the hash algorithm (MD5), the length of the key (1024 bits), the authorities of certification and release (PoliMI), the validity and the holder of the signature.

The validity of a digital certificate is guaranteed by a Certification Authority (CA) (see Figure 10.12). For this reason, a digital certificate is a document that

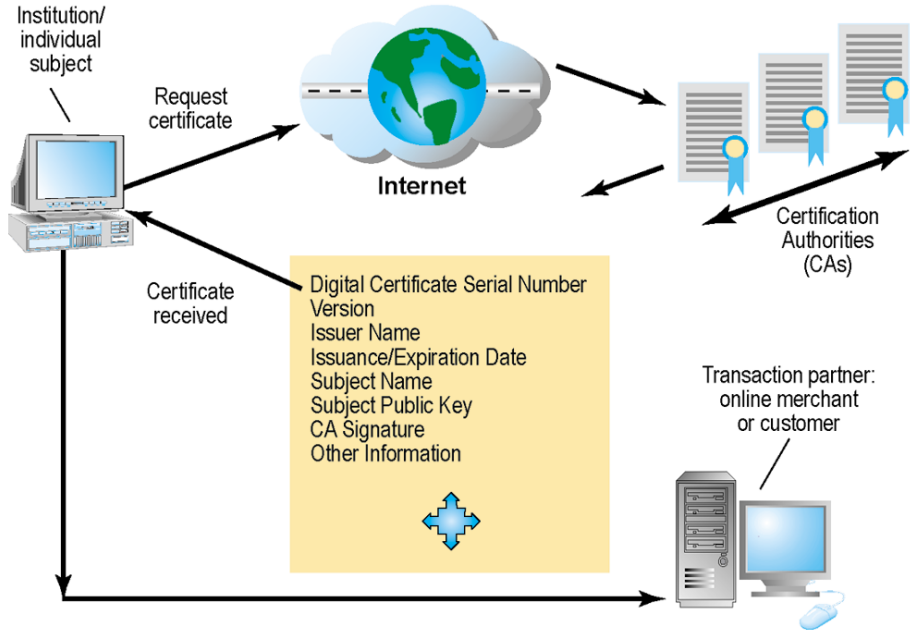


Figure 10.12: Digital certificate and certification authority (source: Laudon)

contains information on who is the owner of the public key, the public key itself, and everything is signed by the CA to ensure that no one has altered the certificate.

In the Certification Authority is officially recognized, the public key can be used to legally sign documents with digital signatures (see Sect. 10.1.6).

To ensure that the certificate cannot be altered, it is protected with the digital signature of the Certification Authority that issued it. When a public key is received through a certificate, it is necessary to check the integrity and validity of the signature: the digest calculated on the certificate (i.e. on the public key and on the data associated with it) is compared with the digest extracted from the digital signature affixed by the CA that issued the certificate.

Creating and verifying digital signatures

The goal of creating and affixing a *digital signature* (or electronic signature) to a data / information is to guarantee the authenticity of the data and identify with certainty their creator. The digital signature scheme appears in Figure 10.13. Digital signature is defined as the encrypted digest of a document (or message), obtained using the private key of the document sender.

The signature is attached to the document; then, signature and document areshipped. In some cases, before shipping, signature and document are encrypted with the recipient's public key who must use his private key to decrypt

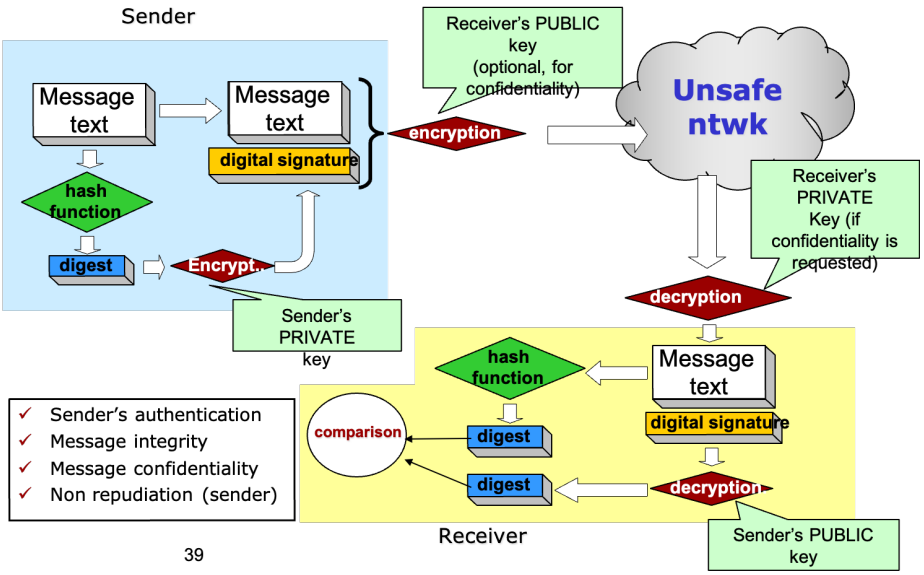


Figure 10.13: Digital signature creation and verification (source: Laudon)

what is received. In general, once the signature and document are received, two actions are performed in parallel: (i) recomputing the document digest; (ii) decryption of the signature with the sender's public key and recovery of the digest of the original document. At this point you can compare the two digests and understand if the document has been manipulated by unauthorized agents.

The digital signature, because it depends not only on the signer's private key, but also on the data through the digest, is different for each different set of data that is signed. The signature affixed to an electronic document therefore cannot be simply copied and affixed to a different document. Finally, it not only identifies who generated the data, but also demonstrates that the data was not changed after being signed and is therefore stronger than the handwritten signature (*non-repudiation* property).

Digital signing takes a long time to process and is typically not used in network processes that require high performance, but only in application solutions where a human agent is directly involved (using SSL, TLS and S/MIME protocols).

Key management aspects and digital certificates

In order to apply encryption algorithms, and sometimes digest algorithms as well, stakeholders need to own and share keys. This aspect is called *key management* (key management) and requires the solution of various subproblems, such as the generation and storage of keys and the exchange of keys (key exchange or key distribution).

The generation of the keys must be done directly by the person who will carry out the cryptographic operations, in exceptional cases (high-security trusted environments) it is permissible to allow the keys to be generated by an entity other than the one that will use them. The cryptographic keys should be random bit strings and therefore a Random Number Generator (RNG) can be used for their generation. The quality of the RNG determines the quality of the security system.

For the storage of cryptographic keys, especially private ones, software or hardware solutions can be used. In the first case, the keys can be stored in an encrypted file protected by a password. As for the hardware solutions, the keys can be stored inside removable hardware devices (e.g. external hard disks) or you can use "active" devices that is able to memorize the keys and carry out the cryptographic operations independently (for example, smart-card, USB token or PC card).

Key exchange is particularly critical in symmetric cryptography as knowledge of the key would compromise communication entirely. One of the most used techniques for key exchange is the OOB (Out-Of-Band) technique in which the key is distributed through a channel other than the one on which the data transit. Alternatively, if the recipient of the encrypted data has an asymmetric key pair, the sender can send the secret key to him by encrypting it with the recipient's public key. Finally, a third party (key distribution center) can also be used.

If the key to be exchanged is of the public type, its secrecy is not important: the problem is to correctly associate the identity of the owner with it. Public keys are distributed through a data structure called *Public Key Certificate* or PKC (Public Key Certificate) of which there are various formats including X.509v3.

Systems that support key generation and management are called PKI (*Public Key Infrastructure*) . These are infrastructures that provide methods and tools to carry out the basic cryptographic functions for a certain community of users, such as:

- issue of public key certificates, after having carried out the necessary technical and procedural checks;
- revocation of public key certificates (for example, following the theft of the private key associated with the public key);
- distribution of public key certificates and information about revoked certificates.

Optionally, a PKI can also provide support for the validation of a digital signature by identifying the time the signature was placed (time stamp), the role covered by the individual who signed (role certification) and the reason why the signature was affixed (signature policy).

The scheme of a PKI is shown in Figure 10.14: the user (seen as the end entity of the scenario) requests a personal certificate from the certifying entity

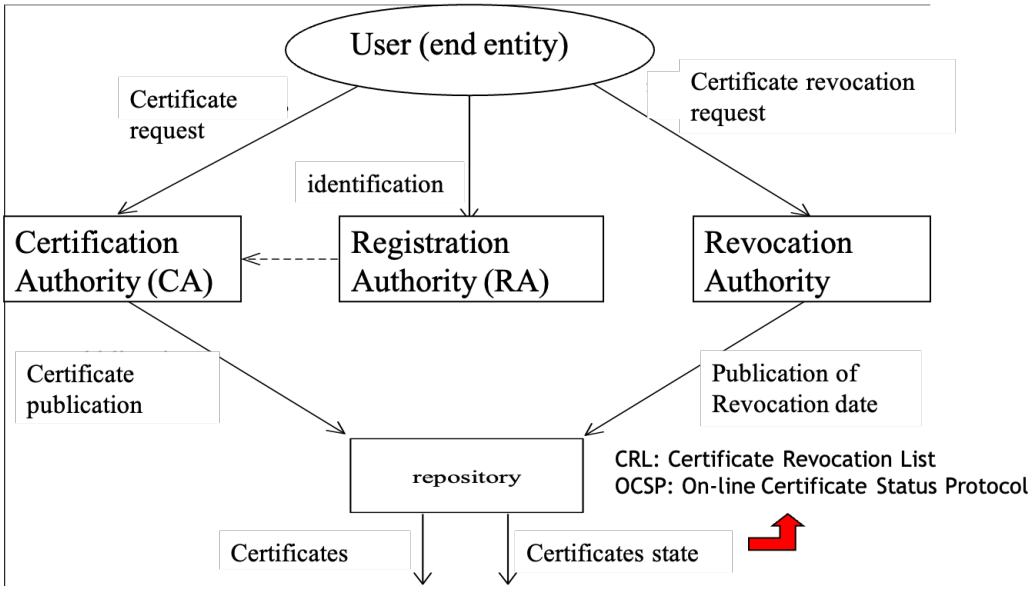


Figure 10.14: Public Key Infrastructure

(CA - Certification Authority) which plays the role of notary, guaranteeing the identity of the parties and also carrying out the role of Registration Authority (RA). Alternatively, the CA can use a trusted third party for the role of RA, in charge of identifying the user, through a specific phase of acquiring the trusted credentials. The certificates are stored in a repository appropriately shared in the cooperation environment, which is accessed to verify the identity of a party (for example, the Internet Banking Server in which sensitive data are being entered). This Repository must also release information on the status of the certificates, which expire or can be revoked at the request of the CA or RA. A Revocation Authority takes care of revoking expired certificates. It is important that the revocation of a certificate is made known in the environment in a timely manner to avoid fraud related to expired certifications.

If this comparison is successful, the certificate is certainly healthy, but it remains to be verified whether the CA that issued it is a trusted CA. For this reason, a list of public keys of trusted CAs is maintained.

Signature verification: an example of verification tool is the following: `https://vol.postecert.poste.it/verificatore/it`. A verification report can be also produced.

10.1.7 Applications

Examples of use of public keys and certificates are the following: exchange of e-mail messages, proprietary applications (for remote control, or to create security modules), access control to hardware equipment (servers, routers, or network

Figure 10.15: E-passport (source [12])

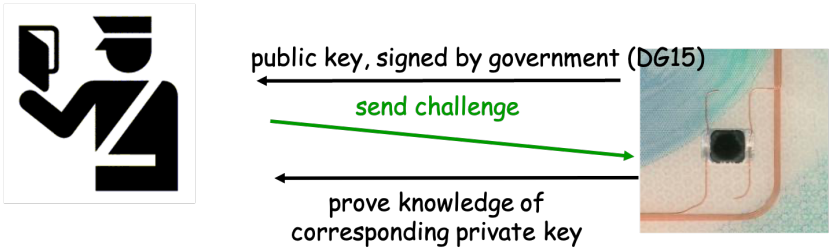


Figure 10.16: Active authentication (source [12])

hardware) , at the application level (for example, for e-commerce applications or to create secure Web sites with the SSL protocol - https) and to obtain software components that are signed, therefore trusted.

Browsers in their settings manage certificates, e.g., by Apple Application Integration CA, VeriSign Secure server CA, Poste Italiane, Chambers of Commerce.

Certificates are also used for client identification (using your own certificate) to access restricted services

E-passport

Public Key encryption is used for E-passports [3]. E-passports contain a RFID with computing capability Fig. 10.15. In particular, it is able to perform encryption and decryption of data. It contains the private key associated to the passport and with the Active authentication mechanism shown in Fig. 10.16. The border-control process is shown in Fig. 10.17.

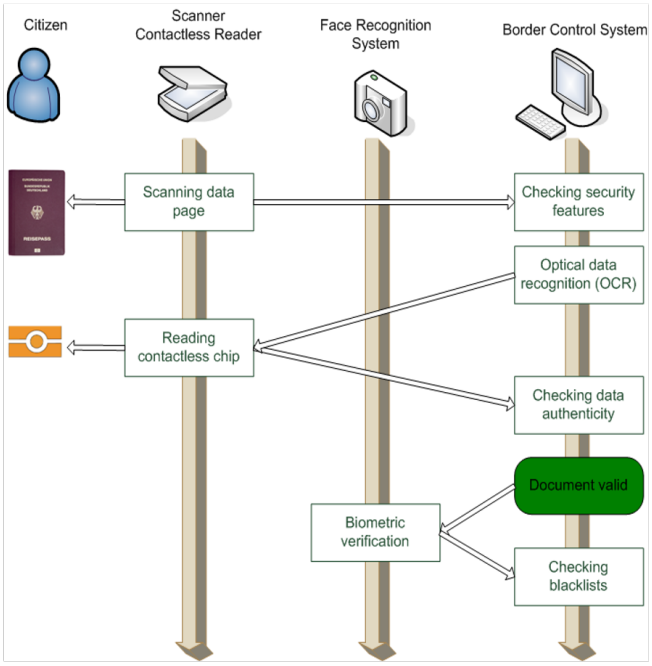


Figure 10.17: Border control (source [12])

Trust-On-First-Use (TOFU)

A party (sender) sends a message and says "this is my public key" the recipient can use this information to confirm that a second message has come from the same source by checking the digital signature or by generating a random number, encrypting it in the putative public key of the sender, sending it, challenging the sender to decrypt the challenge and return it (perhaps encrypted in the public key of the challenger).

Chapter 11

Cybersecurity

Matteo Bregonzio

Cybersecurity for Small and medium-sized enterprises

Politecnico di Milano
9 May 2023

Matteo Bregonzio, Ph.D
Chief Technology Officer, Datrix SPA

Matteo Bregonzio, PhD
Chief Technology Officer @ Datrix
matteo.bregonzio@datrixgroup.com

DATRIX Group
https://datrixgroup.com

Datrix is an Italian SME with headquarters in Milan and offices in Rome, Viterbo, Cagliari and New York

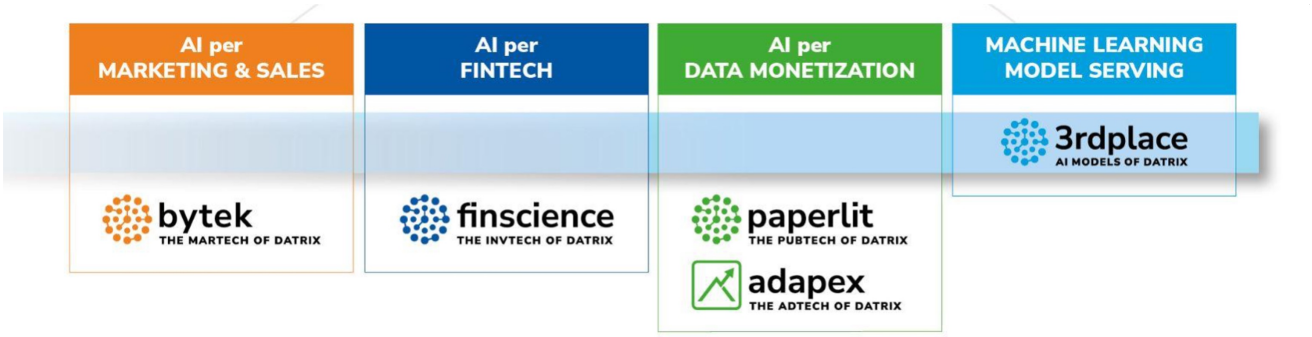
Datrix SpA is listed on Euronext Growth Milan

~100 employees
~€16M turnover in 2022

datrix
AI SOLUTIONS GROUP

DATRIX: Sustainable AI solutions for data-driven Business Growth

SOLUTIONS BASED ON AUGMENTED ANALYTICS AND MACHINE LEARNING MODELS



datrix
AI SOLUTIONS GROUP

DATRIX: Sustainable AI solutions for data-driven Business Growth

Top Customers

euronics · L'ORÉAL ITALIA · Nestlé · ibs.it · LOCAUTO Group · Gruppo Feltrinelli

TIM · italo · sky · BNP PARIBAS · illimity · TECHNOGYM

Topolino · Forbes · PANINI · tag Talent Garden · GRUPPO MONDADORI · CONDÉ NAST

DATRIX: Sustainable AI solutions for data-driven Business Growth

Our technology stack is lean and really on the Open-Source community

Programming languages

Databases

Frameworks and solutions

datrix
AI SOLUTIONS GROUP

DATRIX: Sustainable AI solutions for data-driven Business Growth

Strategic relationships with different universities and collaboration on EU funded projects (Horizon)

Healthcare

Cybersecurity

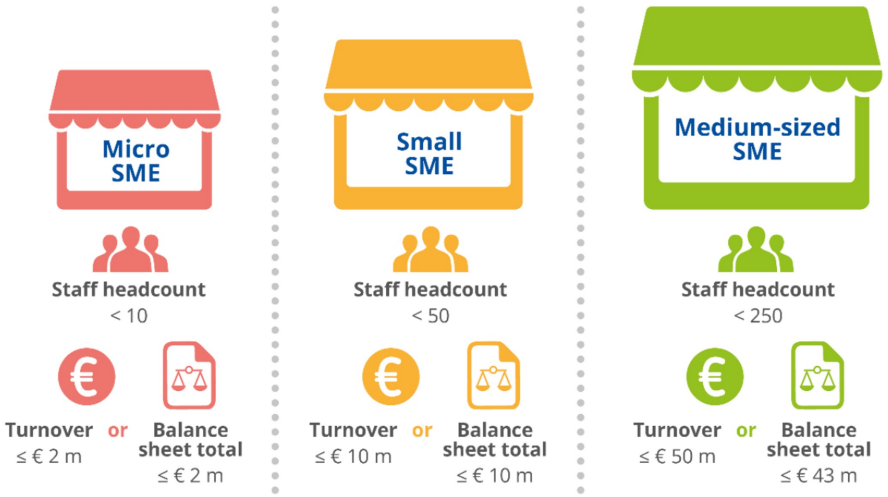
Cybersecurity for Small and Medium-sized Enterprises (SME)

Why SMEs are relevant?

Small and medium-sized enterprises (SMEs) represent 99% of all businesses in the EU.

There were slightly more than 25 million SMEs in the EU in 2018, of which 93% were micro SMEs.

The European Commission, and specifically the EU Agency for Cybersecurity, already highlighted that SMEs are vulnerable and unprepared for cyber attacks. (Report)



Why does cybersecurity metter for SMEs?

1. **The world is getting more digital**
   - Business, banking, healthcare, etc. is all online

2. **Crime is following the same trend**
   - Worldwide ransomware attacks
   - High-profile hacks in the news
   - Phishing emails are more sophisticated each day
   - Professional hacking generates huge profit

3. **New privacy laws and regulations are being enacted**
   - Solutions and services providers must comply with current legislation

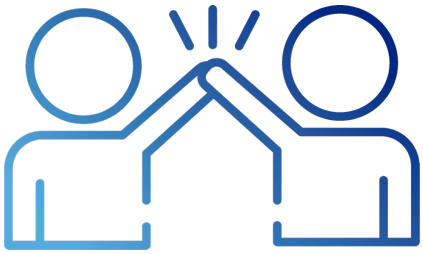
4. **Customers demand for secure and reliable products**
   - Customers' contracts require to follow cybersecurity best-practises.
   - Customers perform regular audit and cheks.

5. **Victims of cyber crime soffer from:**
   - Significant monetary lost
   - Data and assets lost
   - Onerous and slow recovery process.
   - Reputational damage

datrix AI SOLUTIONS GROUP
Cybersecurity

Cybersecurity is a team challenge

- Since the online world is so interconnected, everyone could be a target with their personal or corporate accounts.
- If just one of your accounts gets breached, criminals can use it to breach others
- Criminals may target personal accounts and data to breach corporate ones, and vice versa
- Example: identity theft doesn't just affect an individual; it can affect family members, friends, coworkers, and businesses



⇨ **Training is fundamental** to engage employees and ensure they are aware and capable of spotting suspicious activities.

⇨ **Structured communication** plays a crucial role <<as soon as I noticed something strange, I immediately notify the IT team>>

How likely is that an SME get attacked?

1. On average, an employee of a small business with less than 100 employees will experience 350% more social engineering attacks than an employee of a larger enterprise. (Forbes.com, 2022)

1. 61% of SME were the target of a Cyber Attack in 2021. (strongdm.com, 2023)

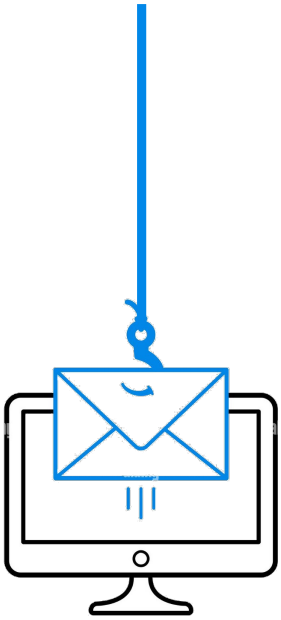
1. SME are more vulnerable to cybercrime, because unlike bigger firms, they are less likely to have teams of IT specialists in place. According to research, over one million UK SME were hit by cyber-attacks in 2018, with an average cost of £6,400, putting many small businesses at risk of closing should the same happen to them. (Zurich, 2020)

1. Phishing most common Cyber Incident faced by SMEs. (EU Agency for Cybersecurity, 2021)
   The report unveils the following challenges SMEs are faced with:
   - Low awareness of cyber threats and unavailability of cybersecurity expertise in the workforce.
   - Inadequate protection for critical and sensitive information;
   - Lack of budget to cover costs incurred for implementing cybersecurity measures;
   - Absence of internal guidelines.

datrix AI SOLUTIONS GROUP
Cybersecurity

What kinds of threat are there?

What kinds of threat are there?



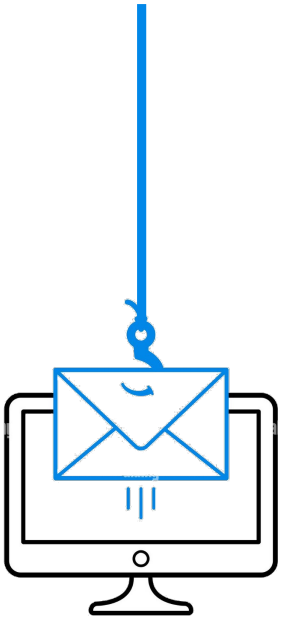
1. **Brute-force attack:** simple yet popular form of attack where the attacker tries multiple login and password combinations in the hope of finding the right one. Example: for the email an employee is using and well-known password such as “juventus”.
   This attack can be executed on multiple entry points including: FTP server, Wordpress admin, SSH server, Remote desktop, etc.

2. **Malware and Virus:** Malware is a catch-all term for any type of malicious software, regardless of how it works, its intent, or how it’s distributed. A virus is a specific type of malware that self-replicates by inserting its code into other programs or PCs. Computer viruses have been prominent since almost the beginning of the commercial internet; the first one was created in 1982.
   Generally, enterprise level anti-malware (anti virus) solutions mitigate this risk (Bitdefender, McAfee, Norton AntiVirus, Sophos, etc.)

1. **Ransomware:** It’s a type of malware that threatens to publish the victim's personal data or permanently block access to it unless a ransom is paid off. While some simple ransomware may lock the system without damaging any files, more advanced ransomware uses a technique called cryptoviral extortion. It encrypts the victim's files, making them inaccessible, and demands a ransom payment to decrypt them

datrix AI SOLUTIONS GROUP
Cybersecurity

What kinds of threat are there?

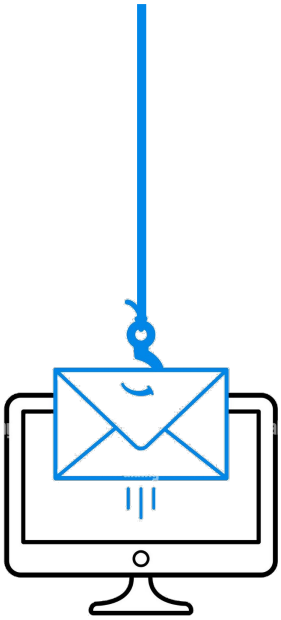


4. **Phishing and spear-phishing attacks**: Phishing can be performed via email, instant message, or text message in order to steal sensitive data. Spear phishing is a specific and targeted attack on a select number of victims, while regular phishing attempts to scam masses of people. In spear phishing, scammers often use social engineering and spoofed emails to target specific individuals in an organization.

5. **Business email compromise:** is one of the most financially damaging online crimes. It exploits the fact that so many of us rely on email to conduct business. In a BEC scam, criminals send an email message that appears to come from a known source making a legitimate request, for instance:
   - A vendor your company regularly deals with sends an invoice with an updated mailing address.
   - A company CEO asks her assistant to purchase dozens of gift cards to send out as employee rewards. She asks for the serial numbers so she can email them out right away.

6. **Social engineering attack:** the attackers don't need to crack your password. Instead, they try to get you to give it to them over the phone / facebook / linkedin / SMS / email / WhatsApp / Telegram / etc by claiming there’s something wrong with your account, and that they’re here to help.

What kinds of threat are there?



7. **New vulnerability are identified on a daily basis (zero-days)**: (also known as a 0-day) it is a set of vulnerabilities previously unknown; until the vulnerability is mitigated, hackers can exploit it to adversely affect programs, data, additional computers or a network.

7. **Cryptomining attacks:** also known as “Cryptojacking” happened when hackers use third-party infrastructure (i.e. Cloud) to mine cryptocurrency without paying for electricity, hardware and other mining resources.

7. **Advanced persistent threat:** long-term attack campaign in which professional hackers attempt to establishes an illicit presence on a network in order to mine sensitive data. The targets of these assaults, which are very carefully chosen, typically include enterprises or governmental networks. The consequences of such intrusions are vast, and include:
   - Intellectual property theft (e.g., trade secrets or patents)
   - National security & militar information
   - Sensitive information (e.g., employee and user private data, credit cards, etc.)

datrix AI SOLUTIONS GROUP
Cybersecurity

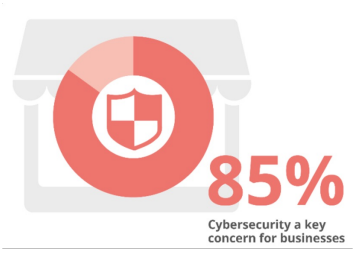
Threat assessment, mitigation and associated costs.

	Datrix Detected	Datrix Victim	Mitigation strategy	Mitigation complexity	Yearly mitigation cost
1. Brute-force attack	YES	NO	Network Firewall, log monitoring, Fail2ban, enforce strong password, password rotation, MFA.	Mid	Cybersecurity expert
2. Malware and Virus	YES	NO	Enterprise level anti-malware solution, make sure it is installed on each and every PCs	Low	€50 per person
3. Ransomware	NO	NO			
4. Phishing	YES	NO	Advanced email service (Google Workspace, Microsoft-365, etc)	Low	€80 per person
5. Business email compromise	YES	NO	Employees training (i.e. 4 houses course once a year)	Low	€15 per person
6. Social engineering	YES	NO			
7. Zero-days	YES	NO	IT Remote Monitoring and Management (RMM), VPN, MFA	Mid	€40 per person
8. Cryptomining attacks	YES	YES	Adoption of software development best practices, log monitoring, principle of least privilege, penetration testing.	Mid	Cybersecurity expert
9. Advanced persistent threat	NO	NO	Log monitoring, penetration testing, 3rd-party audit, MFA, VPN, password rotation.	High	Cybersecurity expert
			Cyber liability insurance	Low	€15K - €50K

Threat assessment, mitigation and associated costs.

SME cyber security yearly costs:

- 185€ per person / year for licenses (antivirus, RMM, email provider) + a cybersecurity expert + cyber insurance
- For a 100 employees SME, It can easily reach €100K / year



Cybersecurity a key concern for the majority of businesses, however:

- Lack of budget allocated to ICT cybersecurity specialists

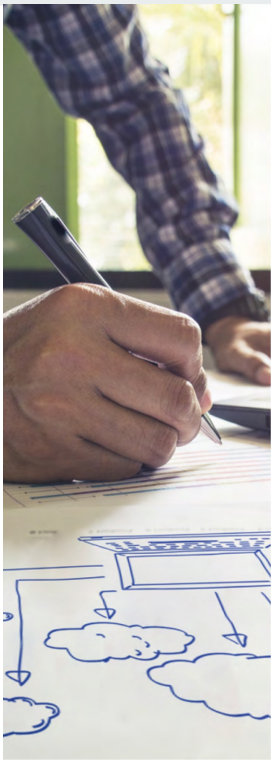
According to a recent survey by the EU Agency for Cybersecurity (link):

- There is an increasing dependency on computers and the internet for all types of SMEs.
- The majority of SMEs use basic security controls such as antivirus protection, backups, firewalls and perform systematic software updates. At the same time fewer SMEs perform security awareness trainings of staff and utilise logging and alerting systems.
- Majority of SMEs do not realize the potential resultant risks posed to their business.

datrix AI SOLUTIONS GROUP
Cybersecurity

Mitigation strategy in practice

Mitigation strategy in practice



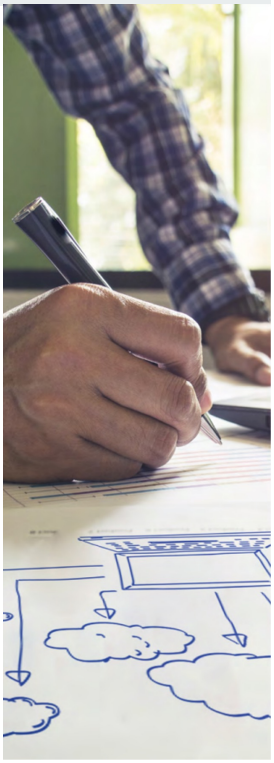
In accordance with the EU recommendations (link), here following are presented list of concrete actions that an SME should take:

1. **Nominate a cybersecurity manager**: he / her should ensure appropriate resources such as time from personnel, the purchasing of cybersecurity software, services and hardware, training for staff, and the development of effective policies.
   - Receive a dedicated cybersecurity budget (i.e. €100K/ year)
   - Publish and disseminate cybersecurity policies for employees on how they are expected to use the IT resources.
   - Perform regular audit involving internal and independent experts.
   - Implement Data Protection Regulation (GDPR) when processing or storing personal data. Ensure that appropriate security controls are in place to protect that data.
   - Publish and disseminate an Incident Response Plan which contains clear guidelines, roles and responsibilities to ensure that all security incidents are responded in a timely manner.
   - Verify vendors, particularly those with access to sensitive data and/or systems, if they meet the required levels of security.

1. **Provide regular cybersecurity awareness trainings for all employees**
   - Ensure they can recognize and deal with the various cybersecurity threats.
     These trainings should be tailored for SME's and focus on real life situations

datrix AI SOLUTIONS GROUP
Cybersecurity

Mitigation strategy in practice



3. Improve physical security: appropriate physical controls should be employed including:

- Anytime a user walks away from their computer they should lock it
- Enable auto-lock function on any device used for business purposes
- Change office's WIFI password every 6 months
- Enforce the use of a password manager, avoiding post-it on the desk

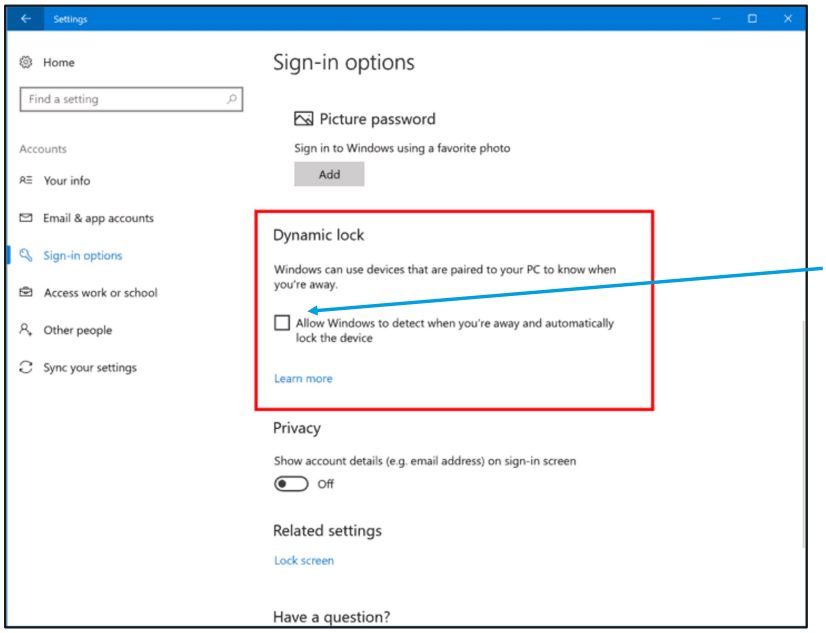
4. Secure access to the systems

- Make sure strong password are enforced (8 characters, letters both uppercase and lowercase, numbers, and symbols)
- Password change every 6 months
- Multi factors authentication
- VPN to connect to your servers or Cloud environment

5. Secure devices

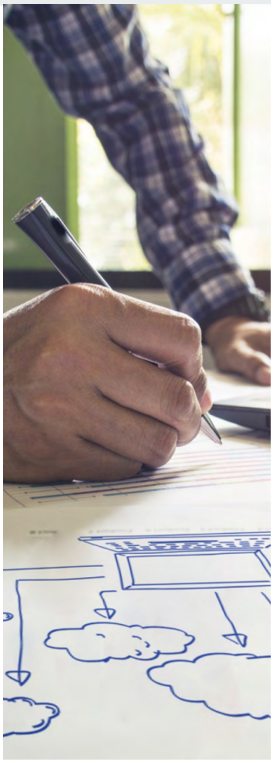
- Install anti-malware (antivirus) software on all PCs and Servers
- Adopt a Remote Monitoring and Management (RMM) solution and install the agent in each PCs. This helps automatically update the OS and installed softwares.

Mitigation strategy in practice - Improve physical security



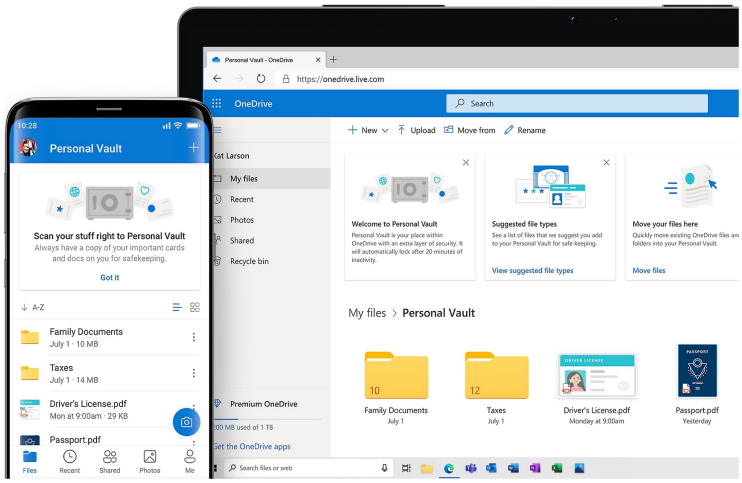
Windows 10

Enable auto-lock function



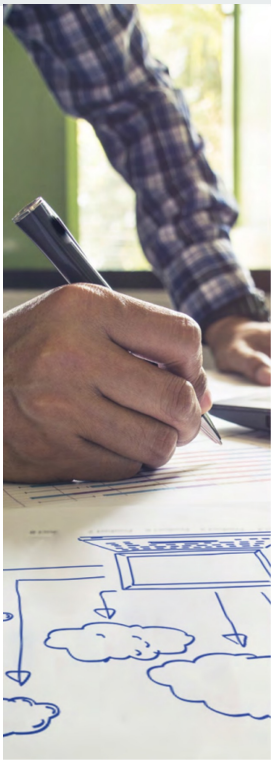
datrix AI SOLUTIONS GROUP
Cybersecurity

Mitigation strategy in practice - Improve physical security

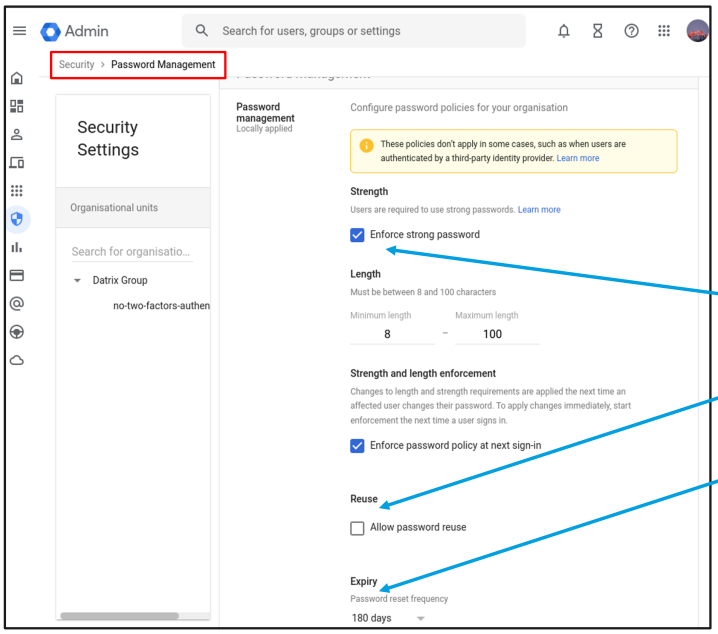


Microsoft 365

Enforce the use of a password manager such as OneDrive Personal Vault



Mitigation strategy in practice - Email security

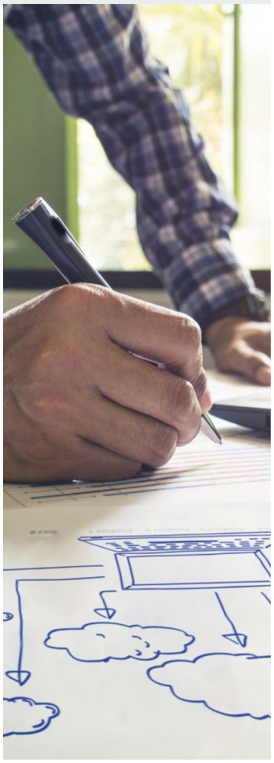


Google Workspace

Strong password

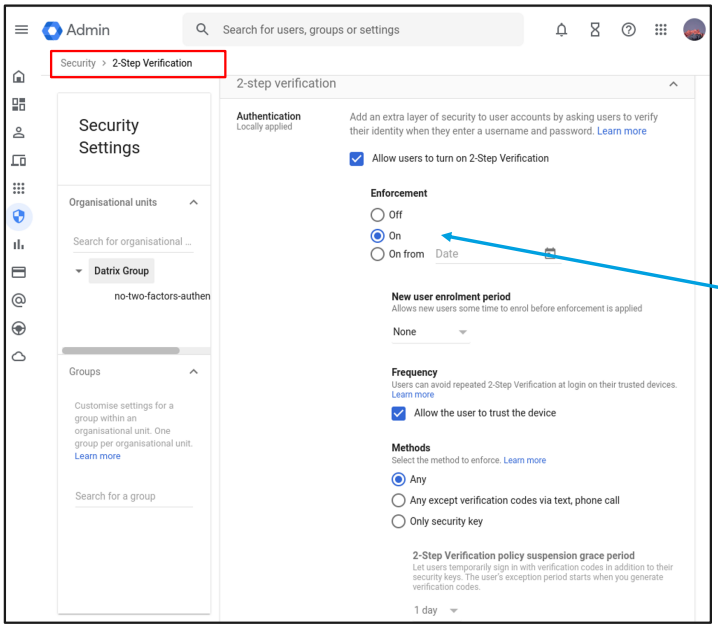
Avoid password reuse

Password change every 6 months



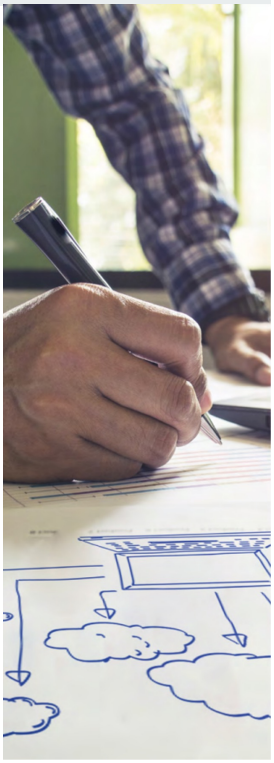
datrix AI SOLUTIONS GROUP
Cybersecurity

Mitigation strategy in practice - Email security

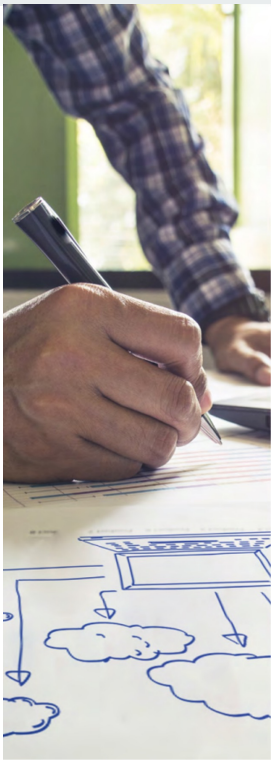


Google Workspace

Multi factor authentication



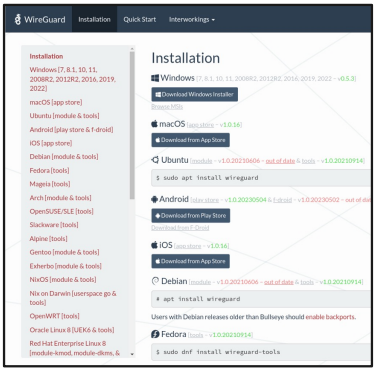
Mitigation strategy in practice - VPN

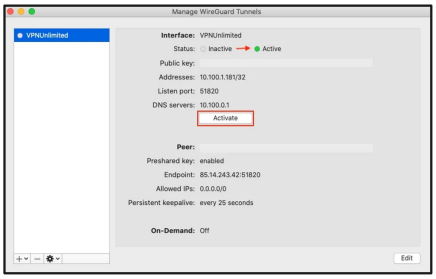


Virtual private network (VPN)

VPN is a mechanism for creating a secure connection between a device (PC, smartphone, tablet) and the office network, using an insecure communication medium such as the public Internet.

There are many VPN solution including WireGuard which is free and open-source. It is easy to install and aviabel for all devices & OS



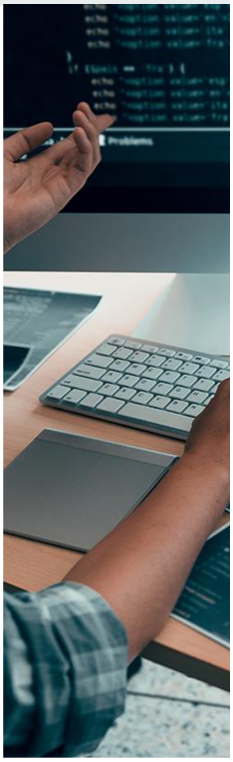


WIREGUARD
FAST, MODERN, SECURE VPN TUNNEL

European Commission

CS-AWARE-NEXT project

Project intro



CS-AWARE-NEXT is a 3-years Horizon Europe research and innovation project that received ~5M€ of funding to research and develop an innovative approach to cybersecurity awareness.

Both Datrix and Politecnico di Milano work at CS-AWARE-NEXT in collaboration with other research centers including University of Oulu (Finland) and University of Vienna (Austria).

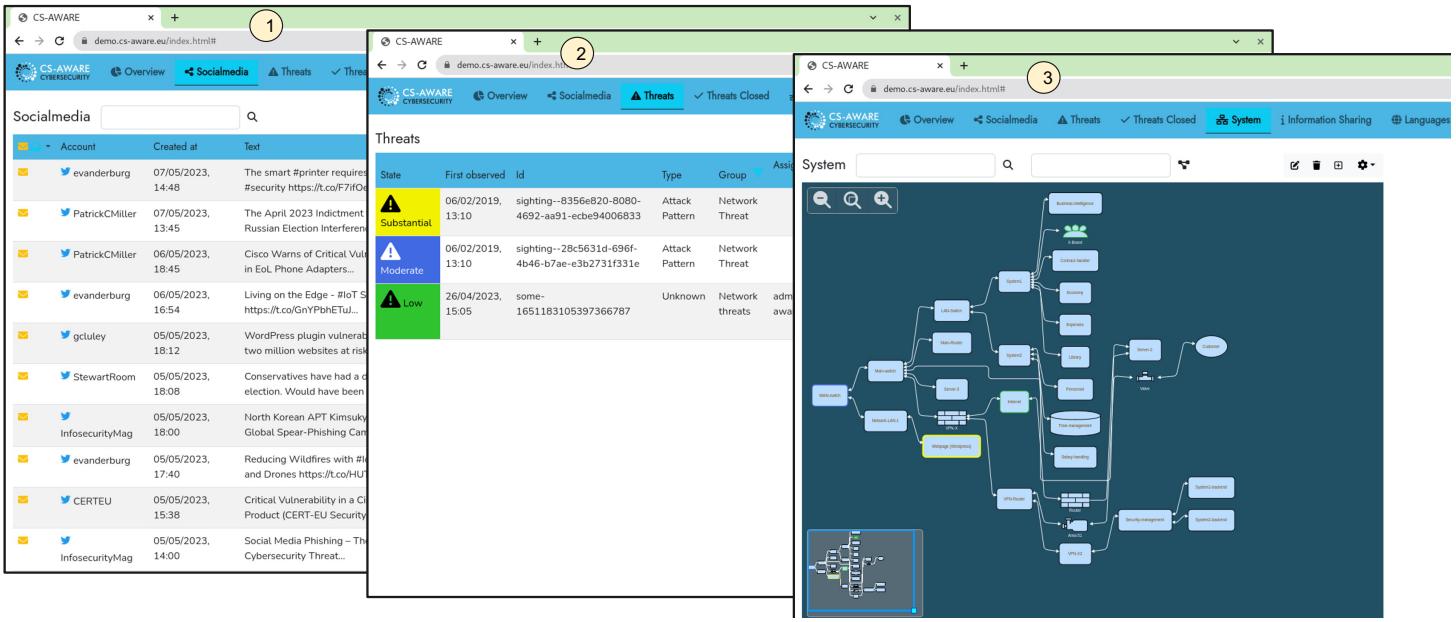
The CS-AWARE-NEXT solution is designed to be used by IT administrator (technical staff) and, differently from traditional security tools such as antivirus or RMM, it aims to provide awareness and a broader understanding of your IT infrastructure security level, anticipating potential issues or vulnerabilities.

Importinalty, **(A)** in line with the EU recommendations, CS-AWARE-NEXT represents also an effective tool for sharing cybercrime information. In facts, sharing of information is crucial to better understand risks factors and mitigate vulnerabilities. **(B)** CS-AWARE-NEXT focuses on a social-technical approach that account not only for technical aspects but also to human behaviors & interactions, cross-department communication, and risk propagation. **(C)** Artificial intelligence is extensively used to analyse multiple data sources and intercept and describe suspicious behaviors.

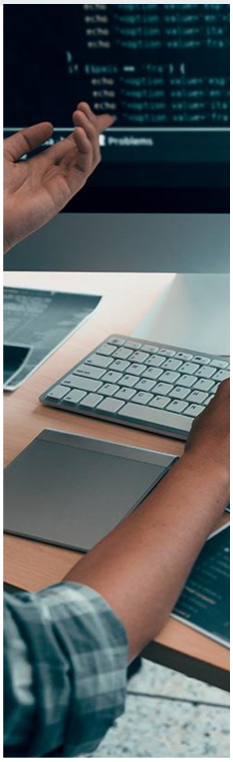
datrix AI SOLUTIONS GROUP
CS-AWARE CYBERSECURITY NEXT

Project intro

CS-AWARE-NEXT web interface: **(1)** collection of social media conversations useful to intercept current threats relevant to my infrastructure. **(2)** Detected vulnerabilities. **(3)** Infrastructure map useful to visualise issues and propagation paths.
Demo Link: https://cs-aware.com/demos/



Project Objectives



1. Provide a cybersecurity management policy framework for organizations to better address the dynamic and constantly changing cybersecurity landscape.

2. Provide a socio-technical framework for local/regional cooperation/collaboration on cybersecurity to better address local supply chain dependencies.

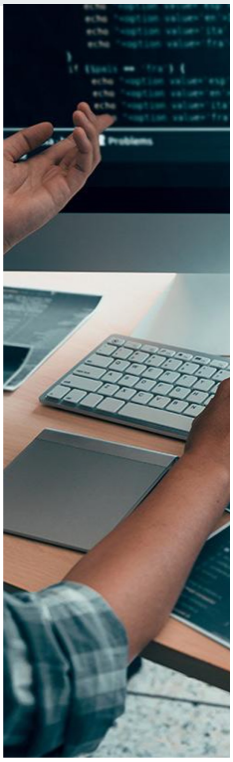
3. Provide a real-time data collection and AI-based analysis framework that is able to collect information from a variety of sources (log files, threat intelligence, social media, ...) and correlate organizational and local/regional information (assets, dependencies, behaviour, ...) with contextual cybersecurity information coming from threat intelligence or social media discussions.

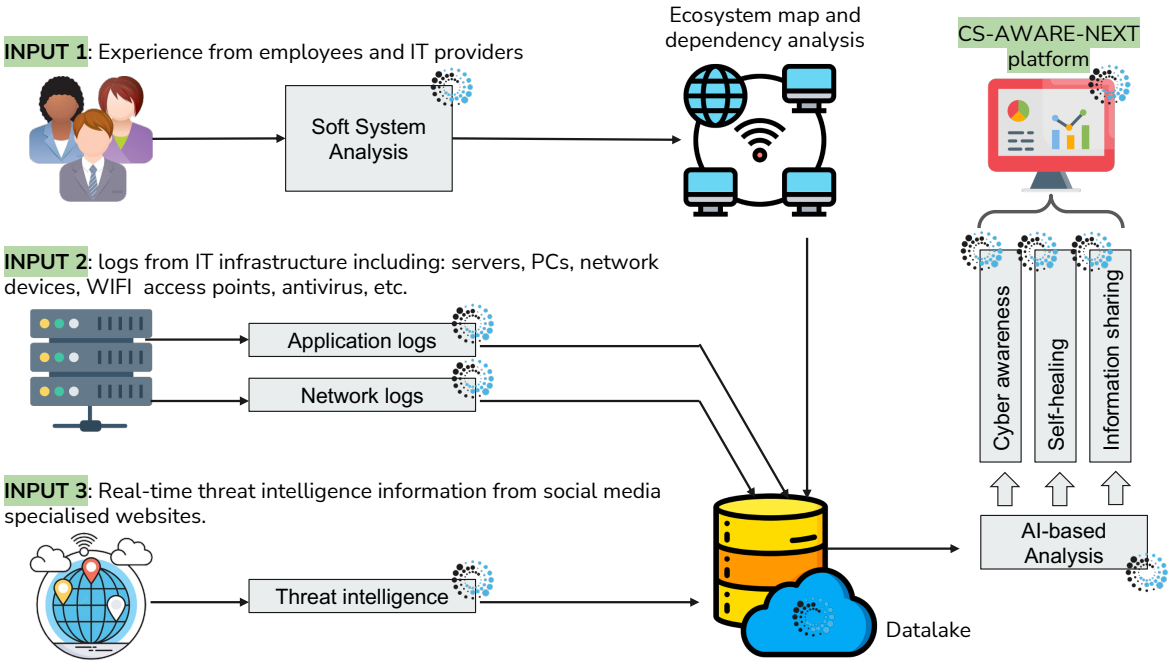
4. Provide a framework for dynamic (real-time) creation and continuous reassessment of disaster recovery/business continuity options relevant to specific organizational or local/regional dependency set-ups to be able to deal with cascading effects.

5. The generation and sharing of threat intelligence based on real-world evidence is one of the core pillars of the multi-level collaborative European cybersecurity framework.

datrix AI SOLUTIONS GROUP
CS-AWARE CYBERSECURITY NEXT

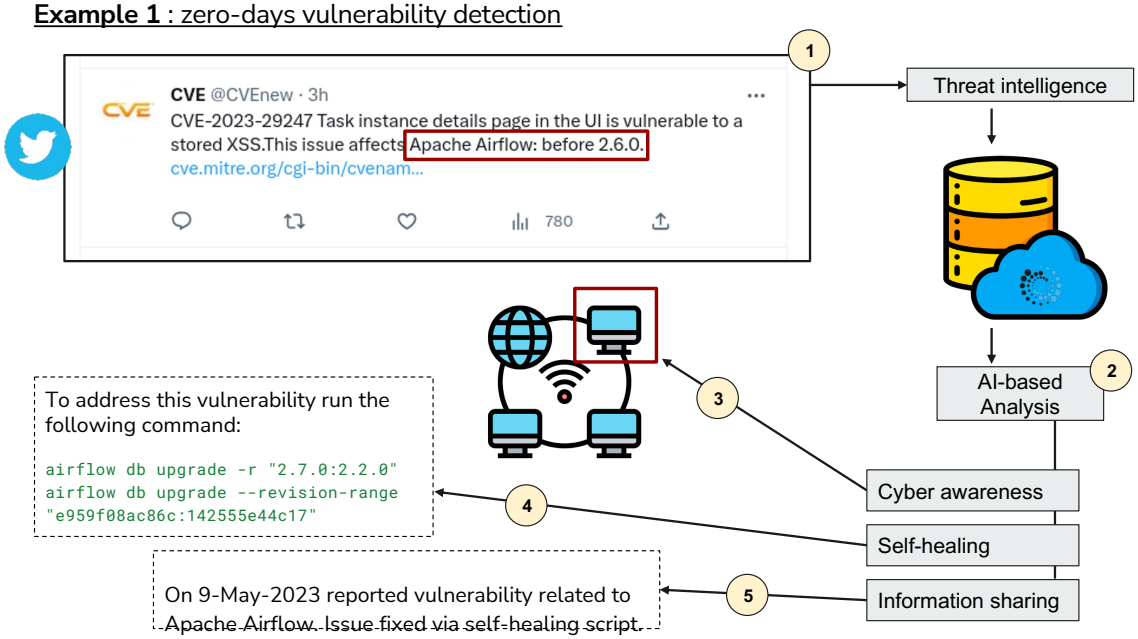
How it works

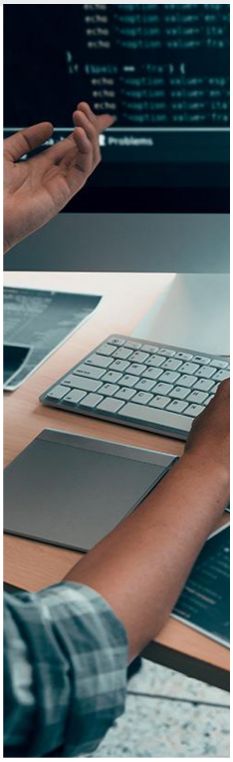




How it works

Example 1 : zero-days vulnerability detection

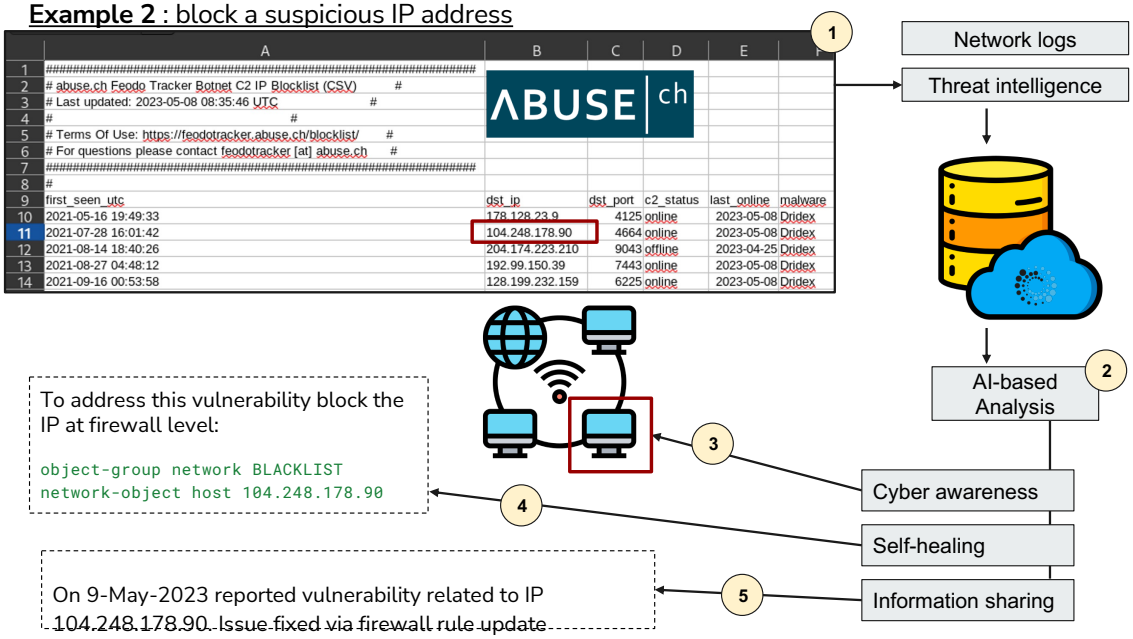




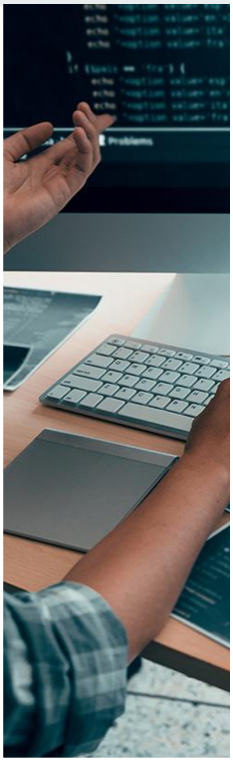
datrix AI SOLUTIONS GROUP
CS-AWARE CYBERSECURITY NEXT

How it works

Example 2 : block a suspicious IP address

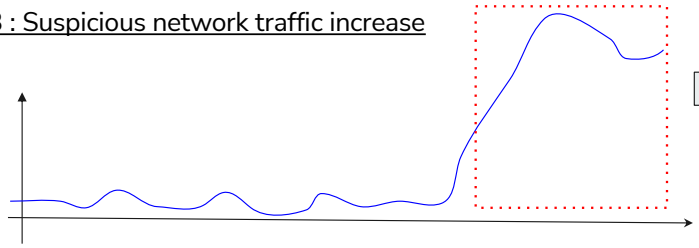


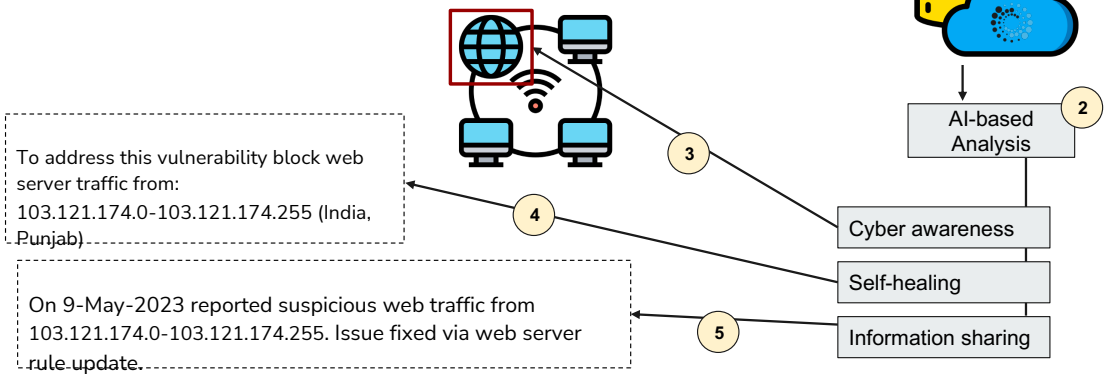
	A	B	C	D	E	F
1	##					
2	# abuse.ch Feodo Tracker Botnet C2 IP Blocklist (CSV) #					
3	# Last updated: 2023-05-08 08:35:46 UTC #					
4	# #					
5	# Terms Of Use: https://feodotracker.abuse.ch/blocklist/ #					
6	# For questions please contact feodotracker [at] abuse.ch #					
7	##					
8	#					
9	first_seen_utc	dst_ip	dst_port	c2_status	last_online	malware
10	2021-05-16 19:49:33	178.128.23.9	4125	online	2023-05-08	Dridex
11	2021-07-28 16:01:42	104.248.178.90	4664	online	2023-05-08	Dridex
12	2021-08-14 18:40:26	204.174.223.210	9043	offline	2023-04-25	Dridex
13	2021-08-27 04:48:12	192.99.150.39	7443	online	2023-05-08	Dridex
14	2021-09-16 00:53:58	128.199.232.159	6225	online	2023-05-08	Dridex

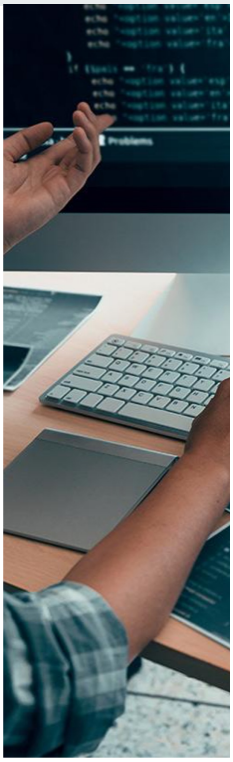


How it works

Example 3 : Suspicious network traffic increase

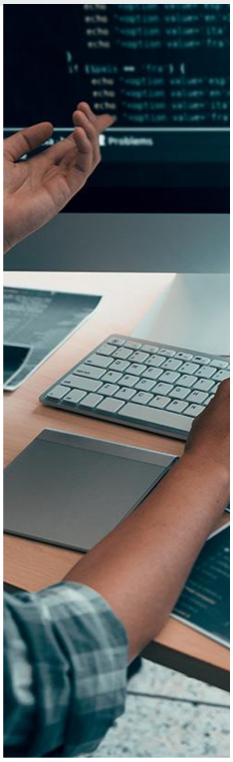






CS-AWARE-NEXT Relevance

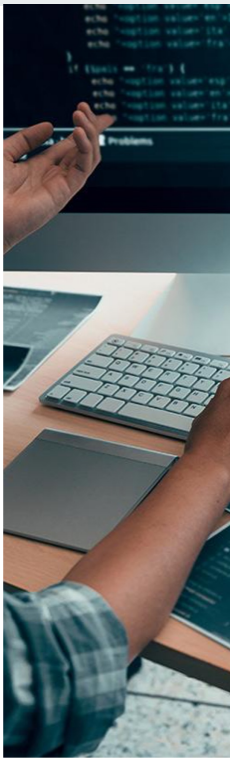
When is relevant ?



	CS-AWARE-NEXT	Mitigation strategy	Mitigation complexity
1. Brute-force attack	YES	**Network Firewall**, **log monitoring**, Fail2ban, enforce strong password, password rotation, MFA.	**Mid**
2. Malware and Virus	NO	Enterprise level anti-malware solution, make sure it is installed on each and every PCs	Low
3. Ransomware			
4. Phishing	NO	Advanced email service (Google Workspace, Microsoft-365, etc)	Low
5. Business email compromise	NO	Employees training (i.e. 4 houses course once a year)	Low
6. Social engineering			
7. Zero-days	YES	IT Remote Monitoring and Management (RMM), VPN, MFA, **log monitoring**	**Mid**
8. Cryptomining attacks	YES	Adoption of software development best practices, **log monitoring**, principle of least privilege, **penetration testing**.	**Mid**
9. Advanced persistent threat	YES	**Log monitoring, penetration testing,** 3rd-party audit, MFA, VPN, password rotation.	**High**

datrix AI SOLUTIONS GROUP
CS-AWARE CYBERSECURITY NEXT

When is relevant ?



1. Monitoring large IT infrastructures composed of devices from multiple vendors and modern and legacy softwares

1. React in a timely fashion to complex attacks and take data-driven decisions

1. Contrary to traditional enterprise level solutions, CS-AWARE-NEXT is more cost-effective and highly configurable

1. Benefiting from information sharing: (1) suggesting resolutions but also (2) receiving suggestion on issues resolution

1. Contribute to create an EU cybersecurity network

Job Opportunities

Current jobs

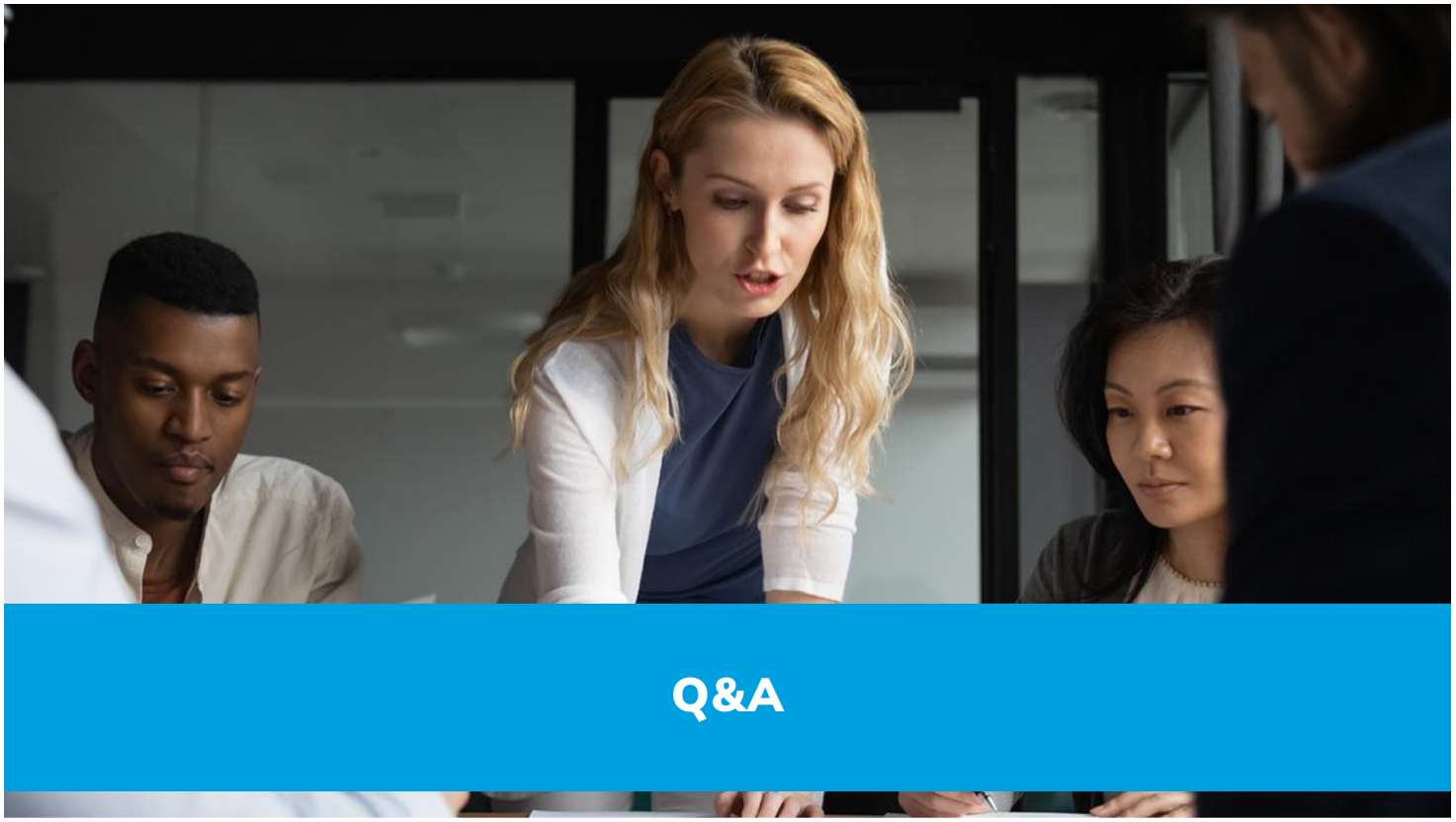
1. Intern / thesis on CS-AWARE-NEXT data analysis - 6 to 9 months

1. Intern / thesis on CS-AWARE-NEXT data collection IT infrastructure deployment - 6 to 9 months

1. Intern / thesis on business development for AIAguard (https://aiaguard.com) - 6 to 9 months

Skills

- Passion about innovation and technology (read news, blogs, test new libraries or frameworks, etc.)
- Motivated to working within a multidisciplinary team
- Demonstrate a security oriented mindset
- Experience with software development and been able to code autonomously (Python, Matlab, Java, go, C/C++, etc)
- Solid understanding of Machine Learning and statistics
- Fluent in English



Q&A

datrix
AI SOLUTIONS GROUP

Thanks !

datrix
AI SOLUTIONS GROUP

datrixgroup.com

finscience
THE INVTECH OF DATRIX

finscience.com

paperlit
THE PUBTECH OF DATRIX

paperlit.com

3rdplace
AI MODELS OF DATRIX

3rdplace.com

bytek
THE MARTECH OF DATRIX

bytekmarketing.it

adapex
THE ADTECH OF DATRIX

adapex.io

Milano
Foro Buonaparte 71
20121 Milano
Tel +39 02 76281064

Cagliari
Largo Carlo Felice 18
09124 Cagliari
Tel +39 02 76281064

Roma
Viale Luca Gaurico 91/93,
00143 Roma
Tel +39 02 76281064

Viterbo
Via dell'Agricoltura 8
00110 Viterbo
Tel +39 02 76281064

New York
27 East 28th Street
New York, NY 10016
Tel +1.718.618.9982

Chapter 12

Blockchain and bitcoins

Marco Comuzzi

Blockchain: a short introduction

Marco Comuzzi
Department of Industrial Engineering
Ulsan National Institute of Science and Technology, Ulsan (South Korea)
mcomuzzi@unist.ac.kr

1.1 What is Blockchain?

One of hardest questions that students ask to a teacher is actually a very simple one: “What is blockchain?”. We begin answering this question in two steps.

The first step (see Section 1.2) concerns understanding the concept of trust and how it underpins the exchange of assets among economic agents. The mechanisms supporting asset exchange can be designed in different ways to support varying levels of trust. Assets can be anything like money, goods, or valuable information. In a digital world, an asset is represented by data in a digital format that prove the characteristics of that asset. There are many kinds of characteristics, but ownership of an asset is obviously a very important one. The exchange of this kind of data represents business transactions, which require a basis for trust. The issue of trust and, more specifically, how it can be created and maintained, is one of the fundamental drivers of blockchain technology. It is also the reason why this technology has been considered revolutionary. To understand the concept of trust and how mechanisms can be designed to enable it, we use the example of different forms that money has assumed through history.

In the second step (see Section 1.3), we move from the concept of trust to the mechanism to create and support it. To do so, we introduce a simple example system that is constructed using the basic functionality associated with blockchain. This is a fictitious online system for e-sports. (online gaming). Online gaming is a booming industry attracting the interest of people of all ages and investors alike. The increasing economic interests around this industry call for mechanisms to improve the trust regarding the execution of games among the stakeholders, like gaming platforms, players, advertisers, and the audience. To keep the example understandable, we consider the game of chess, i.e., a game that everybody has played or at least has heard of at least once in their life. We call this system BC4C (Blockchain for Chess). Using this example, we show in depth what it means to build a blockchain or, in better terms, to design an application based on blockchain that addresses a specific (business) goal. This example also helps us to realize that blockchain implementation relies on several specific technologies, like networking and distributed systems, databases, and, most importantly, cryptography.

1.2 Money, trust, and design choices

Markets, as the locus of economic exchange of goods or services, existed well before the emergence of what we call money today (see Fig. 1.1). In fact, in ancient times, but still even today in many niche scenarios, economic exchange took the form of bartering. A fisherman and a butcher, for instance, would meet on a Sunday morning on the market square of their village to exchange goods: the fisherman would bring fish, the butcher would bring meat. If they wanted and they agreed, they could exchange a fish for a steak. In this way, the butcher’s family could have fish stew on Sunday night, while the fisherman could enjoy a steak before the start of yet another week of hard work at sea.

The essence of such an exchange is the reciprocal trust in the value of the fish and the steak that are exchanged: both the fisherman and the butcher can inspect the fish and the steak that they are about to exchange (for instance, the weight, appearance and freshness) to establish that the fish and the steak have the same value and, therefore, can be exchanged

for one another. Obviously, bartering only works well if two parties want each other's goods and if these goods have the same value. If the butcher wants vegetables instead of a fish, the exchange is much harder: he must assume that he can find a farmer that wants to exchange the fish in return for vegetable produce (which puts the butcher in a bad negotiation position, as he obviously doesn't want the fish).

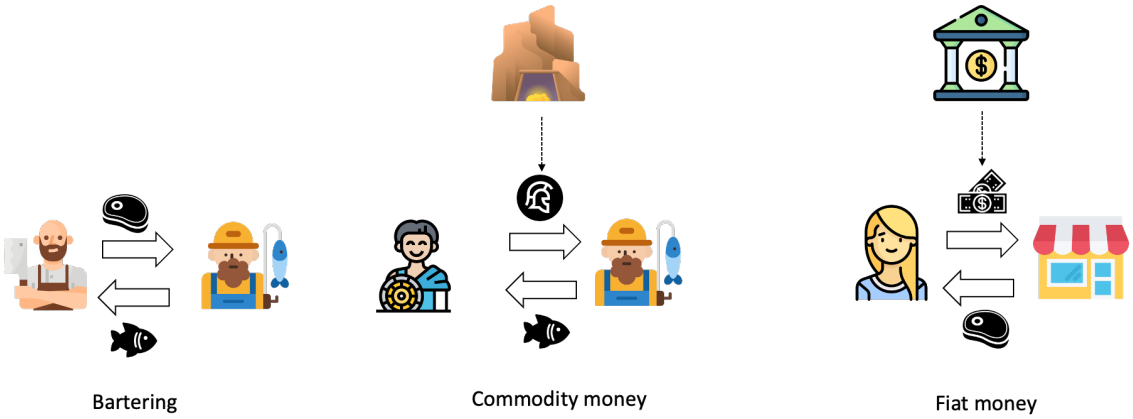


Fig. 1.1 – Different forms of means to regulate value exchange across history

To solve this issue, a more flexible way to regulate an exchange of goods and services emerged very early on in ancient times. It involved the invention of "money" Money can be defined as any item or verifiable record generally acceptable as a form of payment. The initial common form of money is the so-called commodity money. The value of this money comes from the commodity – usually a precious metal - of which it is made. Typical examples of commodity money are the golden and silver coins coined by the Roman empire or sought by pirates across the seven seas. Also in this case, trust is what makes the commodity money a viable mechanism to regulate economic exchanges: the commodity of which money is made is universally trusted to be scarce and durable and, therefore, of a high value to whom owns it. Hence, golden or silver coins can be used to regulate an economic exchange of goods or services deemed of the same value. As everybody wants these coins, we do not have the problem discussed before of a butcher stuck with a fish that he does not want.

Finally, we get to modern times, in which we need even more flexibility to make economic exchanges. We are not used to carry around physical goods (like fish or steaks) to be exchanged for other ones, nor golden or silver coins. Rather, we carry with us paper notes that have an almost null intrinsic value (indeed, what is the actual value of the paper and ink used to print a US dollar note?), but that can still be used in economic exchanges. Or even more extreme, we carry with us cheap plastic cards with a small electronic chip that we do not exchange at all. We simply use these to provide evidence that we own the money required to purchase the goods or services in some bank account. These are the currently well-known debit and credit cards (which would highly confuse the fisherman and the butcher of our first example). While these cards are more convenient and safer to use, they introduce a problem that we do not have with fish, steaks, golden coins or banknotes: the authentication of the owner. Identifying the owner of a fish or a banknote is trivial: the owner is simply the one that has these assets with them when the economic exchange takes

place (obviously these may have been stolen by their current owner, but hey, no system is perfect!). In the case of debit/credit cards, the challenge of authenticating the owner is however deeper. Since the cards are linked to a bank account, every time they are used there must be a mechanism in place to securely and reliably verify that the identity of the owner of the card matches the one of the owner of the bank account to which the card is linked. This obviously requires authentication by the carrier of a card and safe storage of the account details by a bank.

What gives value to this modern form of money? Also in this case, it is the establishment of a trust mechanism. This modern form of money, in fact, is called "fiat" money. Fiat is a Latin word that we can roughly translate into "let it be": money has value because some entity that is trusted by all people "lets this exist". The entity that usually gives value to fiat money is a nation state (or a union thereof in the case of the European Union and the Euro currency). The state prints the notes and mints the coins via its central bank and establishes that these are so-called "legal tender", i.e., they are accepted legally to regulate economic exchanges. In other words, money as legal tender means that a nation state always accepts this money from its citizen for paying taxes. This creates the circle of trust that is required for the circulation of this form of money. People accept to be paid for their labor by their employer using this money because they are able to use it to pay taxes; shops accept it from their customers because, again, they are able to use it to pay taxes; and so forth. Obviously, this mechanism can work only if the nation state issuing the money can be trusted. No one, in fact, is eager to own money issued by countries that are, for instance, politically unstable or engaged in long-standing wars with their neighbors.

A similar trust-establishing mechanism is behind many other forms of 'private' currencies that we commonly use, possibly even on a daily basis. One such example are meal vouchers for a student cafeteria on a university campus. Meal vouchers have value as long as they are exchanged among actors who trust their value. A student may use a meal voucher to have lunch, but also to repay a debt that he has with a friend attending the same university. This works because this friend also trusts to be able to buy lunch with that voucher at the university's cafeteria. However, the same voucher is of no value if used outside the university context. No restaurant outside the university campus accepts the voucher. Consequently, nobody trusts the voucher to have any value outside the university campus. In other words, the university is the 'third party' or the 'central authority' that, similarly to nation states in the case of fiat money, issues and guarantees the value of the meal vouchers.

The newest incarnation of money is that of cryptocurrencies, like Bitcoin. This kind of money does not need to be regulated by a third party, like a nation state or a university. Its value is instead established by the users of the money and by the technological platform that they use to exchange the money. To really understand this kind of money (e.g., its creation, ownership, and exchange), we need to understand more about blockchain. Hence, we will return to this kind of money later in this book. Blockchain is in fact the mechanism behind cryptocurrencies.

To summarize, what have we learned from this brief analysis of different forms of money?

1. That economic exchange depends on trust. Therefore, the value of the means used to regulate an economic exchange is established through a trust-based mechanism.
2. That different design choices can be made regarding the design of different forms of money to establish trust.

We have discussed above four different ways through which people can trust the value of money:

a. The trust in the goods that are physically exchanged in a barter trade (the goods are the money themselves).
b. The trust in the value of the commodity of which the money is made in the case of commodity money.
c. The trust in the institution that issues the fiat money, such as nation states in the case of world currencies or the university in the case of meal vouchers.
d. The trust in the blockchain mechanism through which cryptocurrencies are implemented.

In Tab. 1.1 we show the development of the exchange mechanisms discussed above. It starts with (1) the physical exchange of valuable objects (like meat or fish) which does not need to be regulated by an authority. The development continues with (2) precious coins, which do have an intrinsic value (like the gold used to make them) but are often issued by an authority (like the Roman emperor) that vows for their status or quality, which we call semi-regulation here. The next step is that of (3) physical exchange of fiat money, which requires full regulation by a central authority. When we remove the physical exchange, we get to (4) exchange by credit/debit cards, which also requires full regulation. A development that we have not discussed before is that of (5) community virtual currencies (like currencies in virtual gaming worlds), which we characterize as semi-regulated: there is a regulator, but this is not a formal authority. The final step in the development is that of (6) cryptocurrencies, which do not need a regulation by a central authority nor any physical exchange.

	Non-Regulated by Central Authority	*Semi-Regulated by Central Authority*	*Regulated by Central Authority*
Physical Exchange	(1) Valuable Objects	(2) Precious Coins	(3) Non-Precious Coins Paper Bills
Non-Physical Exchange	(6) Crypto currencies	(5) Community Virtual Currencies	(4) Debit Cards Credit Cards

Tab. 1.1 – Development of different forms of money through time

In the next section, we generalize the issue of trust in regulating economic exchanges to the case of information exchanges. In the modern age, data and information are an extremely valuable asset, increasingly often available natively in digital form. Because of its value,

citizens and organizations strive to protect their personal information. At the same time, the ownership of data and information can accrue immense profits to any organization (for instance for marketing, or generally to gain a competitive advantage). To be effective and efficient, a mechanism for information exchange should address the same issues that we have identified above for money, such as value determination, ownership, or ownership authentication. That is, business actors exchanging information should be supported by a trust mechanism. The most straightforward way to implement trust is again to rely on a central authority. However, the blockchain mechanism can help us to support business actors while exchanging information without relying on a central authority.

As an example, the next section considers the scenario of e-sports, i.e., (professional) competitions using video games. E-sports can be abstracted as scenarios where a number of actors (the players) need to exchange valuable information (the moves that they want to play in a game) to achieve a business goal (winning a game).

1.3 Defining Blockchain by Example: Playing Chess with Blockchain

E-sports are a (professional) form of competitions using video games. They normally involve online multi-player video games often played at a global level. The global market of e-sports was valued slightly above 1 billion USD in 2021 and it is poised to grow above 2.8 billion in 2025 at a CAGR (Compound Aggregate Growth Rate) of more than 20%. The value of this market derives first from marketing activities and game sales. This is due to the sheer amount of interest that e-sports generates in the public, especially in the younger audience. However, a large share of this market also can be generated by gambling, i.e., placing bets on the outcome of games and competitions.

As introduced before, for the objectives of this book, an e-sport online game can be abstracted into the following simple settings: a number of business actors competing with each other to win a game. Playing the game means to exchange information regarding the moves that the players are willing to execute in a game. Regulating this information exchange can be fairly complex, depending on the rules of the game. The rules of an online game in fact are usually more complex than the rules regulating an economic exchange using money. When using money, basically we simply must produce reliable evidence that we have enough money for our purpose (i.e., buying a product or sending some money to a friend). When playing a game, there is normally a (possibly complex) set of rules defining what we are allowed to do in a game.

To continue our discussion, we now need to consider a concrete example. In the interest of clarity, in the remainder we consider the game of chess. We assume in fact that everybody has at least some familiarity with it. Moreover, chess is a game that is regularly played on many online platforms. Note, however, that what we discuss next can be in principle applied to any online game.

The problem that we are facing is the one of supporting players across the world with an Internet connection to play chess online against each other. The players also want to be able to pause a game that they are playing, reconnecting at a later time at their will to resume it. From a trust standpoint, there are two main challenges to be addressed in this scenario:

- To play games, the players must be authenticated: similarly to the problem of money ownership discussed above, there must be a mechanism in place ensuring that one player could not impersonate another player. This creates a basic feeling of trust, with users becoming confident that nobody else except them are allowed to play their own games.
- The state of each game currently played should be stored safely. In this way, it could be possible for the players to pause and resume an existing game. At the same time, it should also be impossible for anybody to modify the state of the games that are being played, except for the two players involved in a game (provided that a player cannot modify the state of a game without the approval of the other one). This creates the trust required by users to pause and resume games at any time. Safely storing the state of the games is similar to safely storing records of money ownership at a bank. In the same way in which we need to trust banks to keep our money safe, players would like that the state of the games that they are playing is stored somewhere safely.

Now, an online gaming platform obviously addresses these two challenges. The platform can authenticate the players in some way, for instance by creating a unique username and password for each of them. The platform can also store the status of each game being played, so that players can disconnect from a game and resume playing at any time. The platform can also ensure that nobody except the two players involved can modify the state of a game, e.g., proposing and executing new moves.

This solution relies on a fundamental assumption: the players must trust the platform. Specifically, they must trust that:

- The platform is always available.
- The platform does not allow the players to cheat, for instance by allowing them to change their moves while other players are disconnected.
- The platform itself does not cheat, for instance modifying the status of an existing games (for instance for the benefit of a malicious gambling cartel!) when the players are disconnected.

Instead of the online gaming platform, can we devise a different implementation of this mechanism to address the trust issues identified above? In the context of this book, the answer is obviously ‘yes’ and it relies, not surprisingly, on implementing a blockchain mechanism. In the remainder, we refer to this mechanism as BC4C (Blockchain for Chess).

The mechanism that we are looking for cannot rely on a ‘central authority’. In this context, we use blockchain to create, by design, the trust that is needed among the players, without the need to create it using a central authority, like the online gaming platform. More in detail, the blockchain mechanism to play chess that we want to design relies on the following fundamental principles:

- Ensuring that the players can agree on the initial state of a game.
- Enabling the recording of the moves played in any game in an immutable database by every player.

- Defining rules to let the players determine which moves are valid, and can therefore be recorded in the database, and which are not.
- Enabling the authentication of the players.

These principles are discussed in detail next.

1.3.1 Defining an initial state of a game

When starting a new game, the players must agree on an initial state of it. In a game of chess, or generally in any online game, any decently knowledgeable player knows what the initial state is. In the game of chess, the initial state is determined by the correct positions of each piece on the chess table. Therefore, the implementation of this step is trivial: we can simply assume that the initial state of the game is always the "standard" one of a game of chess and that players always agree on it. The importance of this step will emerge later, when we discuss how the players could reconstruct the state of games at any point in time.

1.3.2 Defining rules do determine valid moves

The players need a set of shared rules to determine which moves must be considered valid and which others not. Also in this case, the importance of this step will emerge later when discussing the mechanism through which the state of every game can be reconstructed. The implementation of this step is also trivial. Again, chess players with a decent knowledge of the game are likely to know what type of moves are allowed for each type of piece on the board and when a checkmate can be declared. Hence, we can establish the following simple protocol to assess the validity of moves in a game:

- First, a move in an ongoing game is submitted by a player, and
- Second, a move must be approved by the other player before it is considered valid and, therefore, recorded.

Every player stores each approved move in a local database. This database has some special properties. First, new data can be added to this database, but existing data cannot be neither deleted from it, nor modified. Second, the local copies of this database maintained by each player are consistent, that is, they store exactly the same data.

The simple protocol for determining the validity of moves presented above avoids the common issues that could arise in the absence of a central authority. In particular, an invalid move submitted by one player, for instance to gain an unfair advantage, would not be accepted by the other player and never stored by any player in their database[1].

1.3.3 An immutable database to record moves

One important principle of the mechanism governing BC4C is that the database where each player stores the approved moves is immutable. As briefly mentioned above, 'immutable'

[1] Note that malicious players may in principle refuse to accept checkmate or other moves that cause the other player to win. We do not consider this problem in this simple example. In principle, however, this can be solved by allowing players other than the ones involved in a game to certify the validity of the moves. The validity of the moves may even be checked by a computer program in which the rules are simply codified, without the need of any manual approval by the players.

refers to the property of this database to be append-only. We can only add new content to this database. What is already stored in it can neither be deleted nor modified.

Why is such a database required? First, the database stores the moves of every game. In this way, every player can reconstruct the current state of any game simply by replaying the recorded moves starting from the initial state. Second, being immutable, all players can trust that any other player, including the ones against which they are playing games, can reconstruct exactly the same state for any game. Players, in fact, are not able to modify any move stored in the database.

The combination of the three features presented so far (a shared initial state of each game, rules to determine valid moves, and an immutable database to record them replicated by each player) removes the need for a central authority. These features create in fact the trust among players, who can feel free to disconnect and reconnect from an ongoing game. The simple protocol for establishing valid moves, in fact, guarantees (in a reasonable way) that every game is played following the rules of chess. Then, players disconnecting from a game can always reconnect and recreate the current state of any game simply by replaying all the moves stored in the immutable database starting from the initial state. Finally, the immutability of the database guarantees that players cannot modify the moves of any game, including theirs. This for instance removes the risk of players cheating by modifying their previous moves in a game.

Note that, until now, we have not worried about how we can technically build an immutable database, but we simply assume that it is possible to do so. Building an immutable database requires the knowledge of cryptographic tools, which we discuss in Chapter 2. In this chapter, however, it is important to recognize that immutability must be a design property of the database: it is not enough to establish a database usage policy that prevents its users to update or delete existing content. Such a policy, in fact, may be overruled at any time by an administrator. An administrator that could modify any data stored in the database would be an intermediary that the players cannot trust. Instead, the database must physically prevent anybody, not even an administrator, to delete or update existing data in it because of the way in which it is designed and implemented.

1.3.4 Authenticating the players

One final principle of BC4C concerns how to authenticate the players when playing a game. A problem that we have not solved until now, in fact, is how to guarantee that players participating in a game are indeed who they claim to be. An online platform can implement an authentication mechanism easily with usernames and passwords.

How can we authenticate players without a central authority? We do not provide the full-fledged solution to this problem now. As we did above for the immutable database, we simply assume that such a way of authenticating players is technically possible. Conceptually, the solution to this problem lies in switching the perspective. The online gaming platform authenticates the players for ‘sessions’, in which players can possibly play multiple games submitting and/or approving many moves. In BC4C, players are authenticated with ‘every single move’. We require in fact to authenticate every single interaction that a player has with other player, that is, authenticating every move submitted

in a game. Technically, this means that every message submitted by a player to other players must be 'digitally signed'. For now, it is sufficient to know that digitally signing an electronic message means to add to it a little amount of data (i.e., a digital signature) that, similarly to a handwritten signature, allows any other player to identify the sender of the message univocally and undoubtedly.

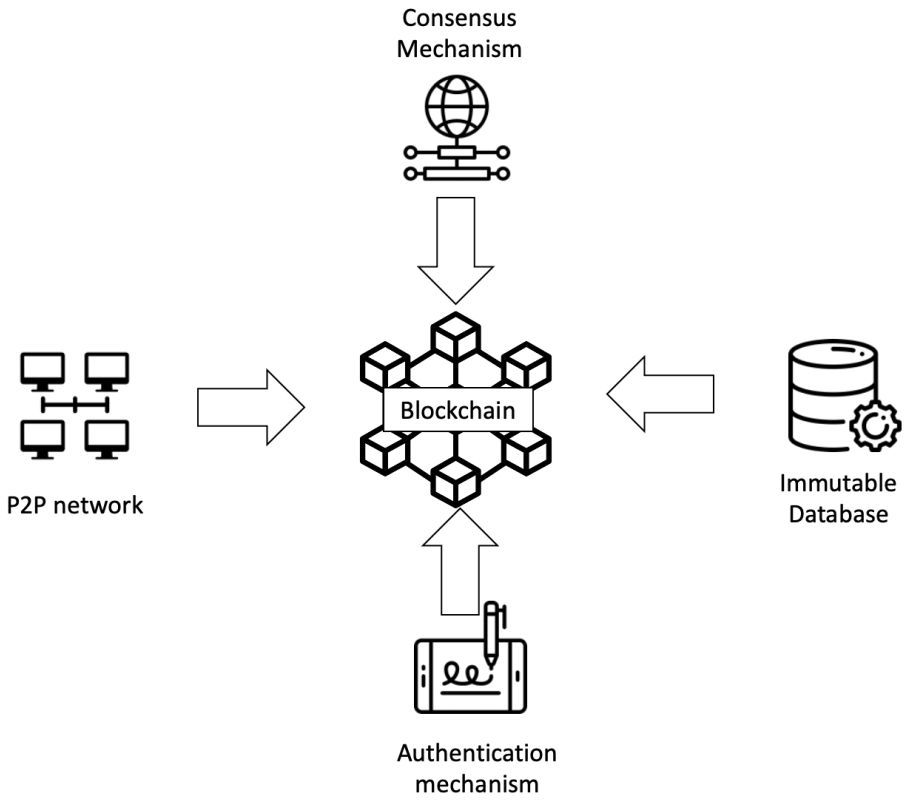


Fig. 1.2 – Basic ingredients of the blockchain recipe

The BC4C mechanism that we discussed in this chapter can be considered a blockchain. Hence, even though we have not yet given a proper definition of blockchain, based on the discussion made thus far we can highlight some of the principal characteristics of a system based on blockchain (see Fig. 1.2):

- It is a system in which there is no central authority, i.e, all participants have the same role, i.e., rights and duties. In BC4C, the only users that exist are the players: they must agree on the initial state when a new game is started, they can submit moves and they must validate the moves submitted by the other players against whom they are playing. A computer scientist calls these kind of participants 'peers' and says that this system relies on a 'peer-to-peer' (P2P) network of users. So, a blockchain involves a peer-to-peer network.
- The system has a set of rules (or a 'protocol', as a computer scientist would say) that are known to all players. These rules guarantee that the players can trust each other when using the system. In other words, the players may not trust each other in principle, but they trust the rules of the system and, through this common trust in

the protocol, they are assured to play the game in a fair way. In the remainder of this book, we will learn to call these rules a 'consensus mechanism'. So, a blockchain involves the definition of a consensus mechanism that guarantees that all participants can reconstruct the state of the system at any point in time.

- The blockchain mechanism is enabled by specific technology. More in detail, we have identified two distinguishing technological features of blockchain systems: immutable databases and digitally signed messages. The former allows every player to reconstruct the state of the games. The latter guarantees that players are authenticated when interacting with other players. More precisely, every interaction that they have with other players is digitally signed.

Now, if we call a message that the peers of a blockchain exchange a 'transaction', then, from a technical standpoint, we can say that *a blockchain involves an immutable database replicated by every peer of a network, who exchange digitally signed transactions*.

1.4 Cryptographic hashing

To understand the inner implementation of an immutable database in a blockchain, we first need to introduce the cryptographic tools used by blockchain. These are the cryptographic hashing functions and the digital signature mechanisms. The former is presented in this section. The latter is presented in Section 4. Section 3 presents the general issue of message encryption. Understanding message encryption is a prerequisite for understanding the digital signature mechanism.

To discuss cryptographic hashing, we first introduce a simple 'mathematical' definition of cryptographic hash function. Then, we discuss the practical meaning of cryptographic hash functions and their applications.

1.4.1 Cryptographic hash functions: definition

A cryptographic hash function *H* is a mathematical function that converts a digital message *M* into a hash value *h* of fixed length (see Fig. 1.3) Usually, the hash value *h* is short. Conversely, the message *M* can be any digital message, such as a short username or an entire book in digital format. Because the size of the input *M* is unbound, the hash value *h* can also be seen as a 'digest' of *M*. A cryptographic hash function that returns a hash of size 'n' bits is referred to as a n-bit hash function. For popular hashing functions, n is usually comprised between 128 and 512 bits.

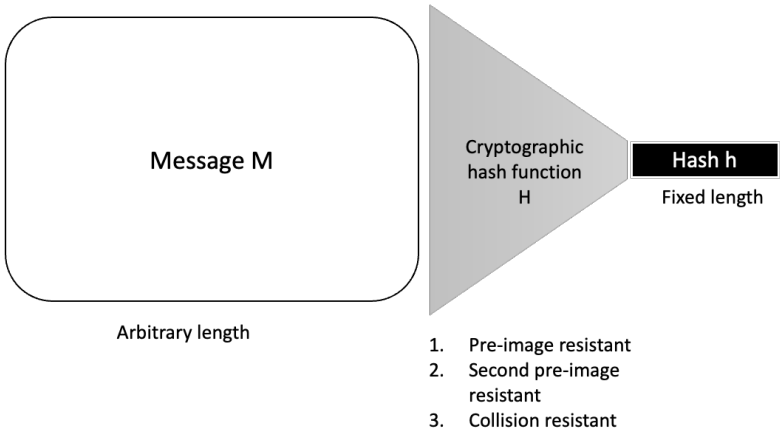


Fig. 1.3: Cryptographic hash function

A cryptographic hash function *H* has three properties:

1. Pre-image resistance: it must be computationally hard to invert the function *H*, that is, to find *M* given the hash *h*.

2. Second pre-image resistance: Given the message *M* and the hash *h*, it must be computationally hard to find a different message *N* such that $H(M)=H(N)=h$.

3. Collision resistance: It must be computationally hard to find two different messages *M* and *N* that map onto the same hash value *h*, that is, for which $H(M)=H(N)$

Note that "computationally hard" in the definitions above means impossible in practical scenarios in the real world. In other words, it must take such a large amount of computational power to, for instance, invert a cryptographic hashing function, that this can be considered impossible to be achieved by anybody in the real world.

The first property establishes that the content of the message *M* cannot be reconstructed from its hash h. Hence, the hash *h* is a digest of the message *M*, but it is not a compressed version of *M*. A compressing function, like the zipped version of a file, must be in fact invertible: given the compressed version, it should be possible to re-obtain the original, non-compressed version. This is not the case with cryptographic hashes: given the hash *h*, it must be impossible to obtain the message *M* that generated *h*.

To understand the second property, let us consider the case in which the hash *h* is required to establish the legitimacy of a certain message *M*. In this context, the second property states that an attacker that obtains both *M* and *h* cannot substitute *M* with a different message *N* using the same hash *h* as a proof of legitimacy.

The third property defines the practical uniqueness of a hash (a.k.a. 'collision avoidance'). In other words, the function *H* maps a message *M* to a unique hash value *h*. Combining the second and third property, a hash h can be seen as a unique identifier of a message *M*. No two messages *M* and *N* should in fact map to the same hash *h*. Even if only one bit in the message *M* changes, then the hash *h* of *M* also changes.

Regarding specifically the third property, it must be noted that it is impossible to design a cryptographic hash function *H* that guarantees the complete 'collision avoidance'. Demonstrating this is trivial, even for readers who are not computer science experts. This property derives from the fact that the size of the input *M* is unbounded and, consequently, infinite. Since the domain of the function *H* is unbounded, it is impossible to guarantee that, for any cryptographic function *H*, collisions (i.e., two messages mapping to the same hash) may never ever be found. When designing *H*, the best we can do is to make such collisions 'highly unlikely' or, as said above, virtually impossible in practical scenarios.

1.4.2 Applications

Cryptographic hash functions are widely adopted in blockchain. At this stage in the book, however, we do not know enough about blockchain to discuss this kind of applications. Nevertheless, we can discuss other well-known applications of cryptographic hash functions in the context of proof of file integrity and encrypted passwords. Applications of this kind have been in place for many years. They underpin the functionality of the Internet as we use it every day.

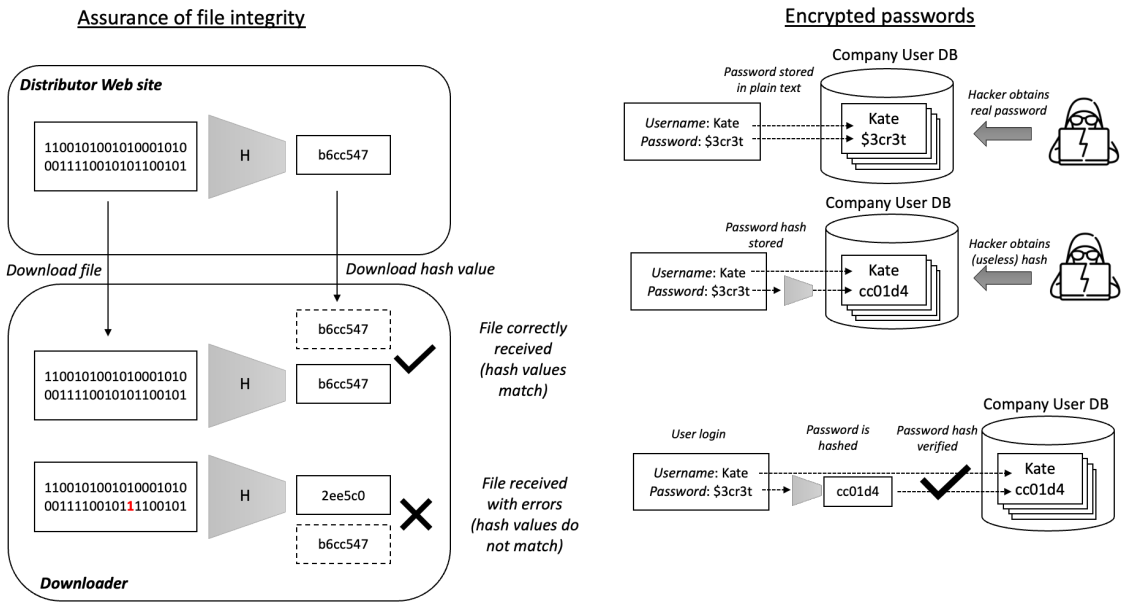


Fig. 1.4 – Applications of cryptographic hash functions

One problem that cryptographic hash functions can solve is that of proving the integrity of a file that we download from the Internet (see Fig. 1.4a). This is particularly relevant when downloading and installing software applications on a computer. A program installing a software on a computer usually requires high-level privileges, like the ability to write on the hard disk and to modify the configuration settings of the operating system. A "man-in-the-middle" may then have an incentive to modify the content of what we are downloading to achieve some malicious objective. For instance, it may modify the content of the download such that, instead of downloading and installing a simple editor for our photos, we would instead install a computer virus. The virus may then take control of the files on our computer, asking for a ransom in Bitcoins (!)

How can we avoid this situation using cryptographic hashing functions? The key lies in distributing also the hash of the content that has been downloaded. The software distributor would make available, besides the file to download, also its hash *h*, calculated using a (well-designed) hashing function *H*. The downloader, after finishing the download, runs the same function *H* on the file downloaded. In this way, they can verify whether the hash obtained matches the one published by the distributor. If that is the case, then the downloader can rest assured that the file downloaded is exactly the one published by the distributor. If, in fact, even only 1 bit of the file was maliciously modified during the transmission, the hash obtained by the downloader will not match the one published by the distributor. If not, then the downloader should not install the software downloaded (and probably also notify the distributor that something is wrong!). Note that while the attacker in the middle could modify the file during the download, the hash can simply be published by the software distributor on a Web page for the downloader to verify. The hash is in fact a relatively short sequence of bits.

A second problem that is addressed by cryptographic hash functions is encrypting user passwords (see Fig. 1.4b). We maintain accounts (usernames and passwords) on dozens of Web sites. These sites obviously need to store our password in some database in order to verify it whenever we login. However, does this mean that, for instance, a database administrator of a Web site on which we have an account can actually see our password? Malicious administrators may obviously exploit this privilege. For instance, they may collect all users passwords, selling them to hackers or using them directly for some illicit activities. Thanks to cryptographic hashing functions, the answer to this question is obviously 'no'.

A Web site in fact stores our password in an 'encrypted' format. This means that (i) the password is not stored in its original format (the 'plaintext'), but (ii) it can still be verified whenever needed. In practice, this is done by storing the hash of a password obtained using a cryptographic hash function instead of the actual password. The hash of the password cannot be inverted to obtain the actual password (thanks to the 'pre-image resistance' property mentioned above). Whenever a user input the password, this can be verified by calculating its hash and checking whether this matches the one in the Web site database.

1.4.3 Asymmetric Encryption

Asymmetric encryption is implemented by cyphers in which different keys are used for encrypting and decrypting a message. Given the complexity of this kind of cyphers, it is not possible to provide a concrete example of them in this book. References to concrete asymmetric systems are given at the end of this section. The objective of this section is to present asymmetric encryption conceptually. Specifically, we want to highlight how it can support more complex cryptographic applications than symmetric encryption.

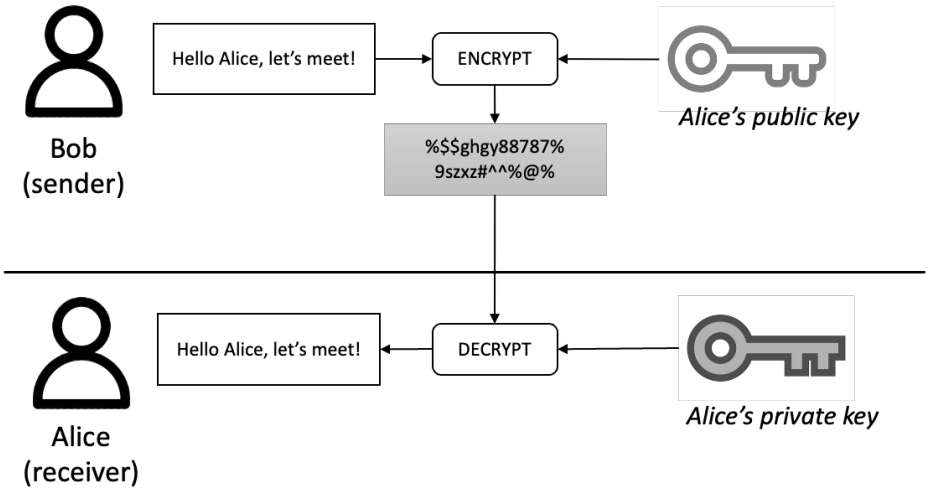


Fig. 1.5 – Asymmetric encryption (public key encryption)

The basic mechanism of asymmetric encryption is depicted in Fig. 1.5. It involves two keys. These two keys are mathematically related so that one is used for encrypting messages and the other one for decrypting them. The keys are also assigned another property, i.e., whether they are 'public' or 'private'. A private key is such that it should not be distributed by its owner. It must be always kept secret by the owner. A public key can be distributed by its owner to anyone who needs it. Note that in symmetric encryption a key is always private. For instance, neither Caesar nor his lieutenants and soldiers should ever make the value of the key public when using a Caesar cypher. If an enemy gets hold of the value of the key, in fact, they could decrypt any message exchanged by them, or even encrypt fake commands as if they were issued by their general.

In asymmetric encryption, a user (Alice in Fig. 1.5), who wants the world to be able to communicate with her, makes her public key available. Now, Bob can use Alice's public key to encrypt a message that he wants to send to her. Attackers in the middle may eavesdrop Bob's message. However, they will obtain only an encrypted message that they are not able to decrypt. Alice is the only one who can decrypt the messages encrypted using her public key, thanks to her private key. As mentioned before, secure communications between Alice and the world (including Bob) are guaranteed as long as nobody except Alice knows her private key.

Asymmetric encryption improves the symmetric one by reducing the number of actors required to know a private (secret) key. With a symmetric cypher, everyone involved in the communication must know the value of the key, which must remain secret to attackers. In asymmetric encryption, only the private key of the receiver must remain secret to the attackers. If Julius Caesar knew about asymmetric encryption, he could have written his public key on every road milestone across Europe!

1.5 Digital signatures

The last cryptographic tool that we must introduce to understand blockchain is the digital signature mechanism. The problem addressed by digital signatures is the one of authenticating the digital messages between a sender and a receiver. Authentication is defined by the following two properties:

- The receiver must be able to verify undoubtedly the identity of the sender of a message.
- The sender must not be able to repudiate a message that they sent to a receiver.

These two properties are the same that we expect from other authentication mechanism that we use in the real world. We sign a private letter (when we still send it instead of an email!) to make sure that the receiver knows undoubtedly that we (and not somebody else) wrote it. Similarly, our signature is required on important documents often for a matter of non-repudiation: by signing a document, we acknowledge that we read it and we cannot repudiate that we know what is written in it in the future. Authentication also exploits other means than signatures. A pin number for a debit or credit card is a typical example of a private key. We must keep it secret, and it identifies us undoubtedly as the owner of the card associated with it.

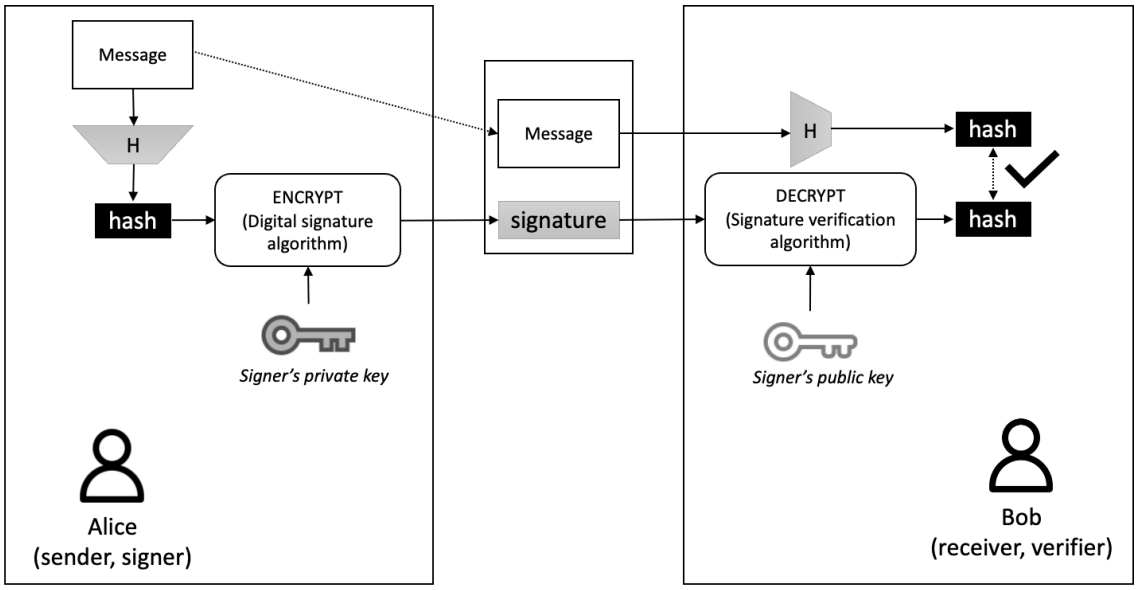


Fig. 1.6 – The digital signature mechanism

Digital signatures implement an authentication mechanism similar to unique pin numbers in the case of digital communications. Fig. 1.6 presents the digital signature mechanism. It combines asymmetric encryption and cryptographic hashing. Alice (the sender or, more generally, the 'signer' of a message) wants to send a message to Bob (the receiver, but also the 'verifier' of the signature in this context). Alice wants her message to be digitally signed. In this way, upon receiving the message, Bob will know for sure that Alice is indeed the sender of it. First, Alice calculates a cryptographic hash of the message that she wants to

send (step 1). Then (step 2), this hash is encrypted by Alice using her private key. The hash of the message is its digital signature. Thanks to the properties of cryptographic hashing, it is in fact practically unique and it can only have been generated from this particular message sent by Alice. Any other message, which differs for at least 1 bit from the one sent by Alice, has in fact a different hash. Moreover, it could only have been generated by Alice using her private key. In Step 3, the message and its digital signature are sent to Bob. After receiving the data, Bob does two things. First (Step 4), he calculates the cryptographic hash of the message. Then (Step 5), he decrypts the digital signature using Alice's public key. Finally (Step 6), he verifies that the hash of the message matches the decrypted digital signature.

The mechanism described above ensures that:

1. The message has not been modified while in transit. If it had, in fact, the hash calculated by Bob would not match the digital signature attached by Alice to her message.

2. Bob can establish undoubtedly that Alice is the sender of the message. The digital signature, in fact, could have been produced only by Alice, using her private key.

3. Alice cannot repudiate the fact that she sent the message received by Bob. The digital signature, in fact, could have been generated only by her.

Note that, when compared to the asymmetric encryption discussed in the previous section, in the digital signature mechanism the roles of the private and public key are switched. The sender uses the private key to generate a part of the message (the digital signature). The receiver uses the public key of the sender to verify the authenticity of the digital signature.

It is also important to understand that the digital signature mechanism is orthogonal to message encryption. In Fig. 1.6, the message is digitally signed, but not encrypted. This means that Bob can verify that Alice is the sender of the message, but a malicious attacker can eavesdrop the message in plaintext while in transit. The digital signature mechanism of

In practice, digital signatures in the digital world work very similarly to wax seals in the (ancient) real world. High authorities, such as queens and kings, or the pope, used to send around the world their communications enclosed in envelopes sealed using a wax seal. A wax seal was supposed to be unique to the sender and hard to forge. For instance, it could be generated by imprinting a royal or papal ring, which was virtually impossible to forge, into warm wax. For the receiver, the wax seal was a guarantee of the identity of the sender. The message enclosed in a wax sealed envelope, however, may or may not have been encrypted using some secret code.

1.6 What is blockchain (again)?

After having introduced a basic cryptography background, we can finally answer the question that we started with: "What is blockchain?". Blockchain is a mechanism, implemented through several technologies, to build systems that enable their users to exchange digital information (representing assets, like money, physical services and products, or valuable information) in a 'trustless' way. Trustless means that the users of a

blockchain system may not trust each other (and we should never assume that they do, unless explicitly specified). Trust among the users is guaranteed instead by the blockchain mechanism, which underpins the system that they use to exchange assets. It is fundamental to understand that the trust of the users in the blockchain technology derives from its design features, such as the immutability of the data stored in it or the consensus mechanism. In other words, the trust is created by intrinsic features of the blockchain technology that the users cannot change or adapt to their specific needs.

From a technical standpoint, blockchain combines several technologies. Some of these, like digital signatures and cryptographic hashing, belong to the discipline of cryptography and have been introduced earlier. Other technologies, such as peer-to-peer networking and immutable databases, are discussed next.

Through the use of technology, the blockchain mechanism enables the implementation of blockchain systems. These are distributed information systems that use blockchain at their core to create trust among their users. Blockchain systems are the foundation of cryptocurrencies, which means that all cryptocurrencies have a blockchain system at their core that users must use to exchange the cryptocurrency tokens. However, blockchain systems are not limited to cryptocurrencies. They support the implementation of information systems in several different scenarios, ranging from shipping containers management, to monitoring the food provenance or the diamond trade, or energy exchange in private microgrids.

1.7 Blockchain as a P2P network

A blockchain system involves a peer-to-peer (P2P) network of computational nodes connected through the Internet. A P2P network is such when all its nodes are the same, that is, they have the same rights and they get access to the same functionality. File sharing networks, such as the BitTorrent protocol, are typical examples of P2P networks. Users of a file sharing application can download files hosted by other users of the network and can host files to be downloaded by others. A P2P network assigns the same rights to every node, but it does not enforce the exercise of any of such rights from any of its users. For instance, in a file sharing network every user has the right to host files that others can download, but many users do not use this right. They simply connect to the network to download files, but they do not host files that others can download. Other users act mainly as content providers, i.e., they provide files for others to download but they do not use the network to download files. Other users do both. Similarly, the nodes of a blockchain system are usually granted the same rights, but it is up to them to exercise these rights. In a blockchain system implementing a cryptocurrency, most nodes use the system to exchange the cryptocurrency tokens, but only few of them, implement specific functionality required to execute the consensus protocol.

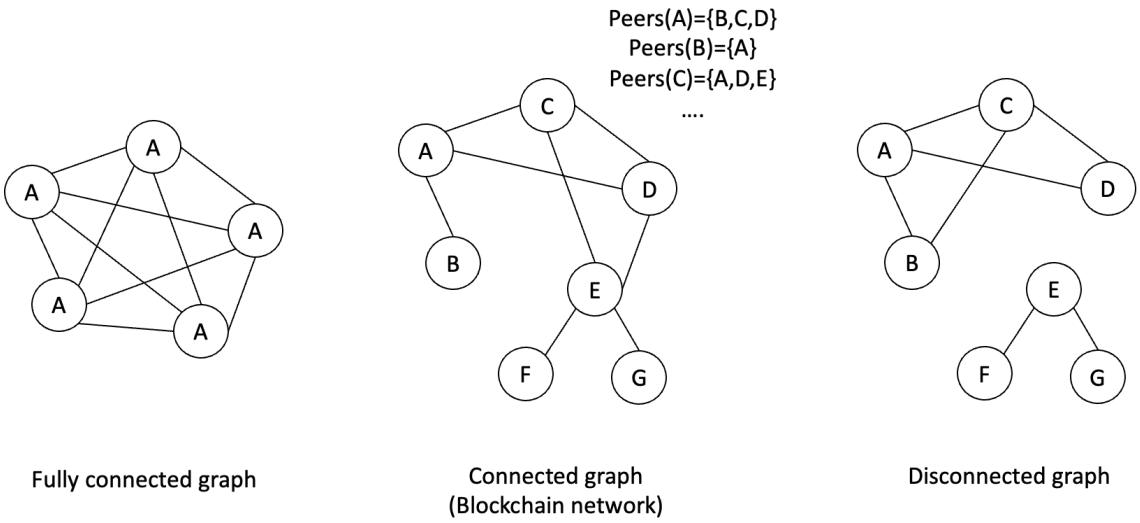


Fig. 1.7 – Connectivity in graphs and peer-to-peer networks

Users connect to a blockchain network using a computer connected to the Internet and by running a specific software application. The functionality of this software is specified by a consortium, foundation, or other organization(s) who oversee the development and management of a blockchain network. As in any P2P network, when connecting to a blockchain network, a user normally does not directly connect to all the nodes that are already connected to the network. In other words, a computer connected to a blockchain network does not know an Internet address for all the other computers connected. Instead, a new user connects to a limited set of other users. These are normally chosen based on geographical proximity. The blockchain network, however, is built in such a way that there is always a path, starting from a given user, to reach any other user. Therefore, the nodes of a blockchain network form a connected graph, but a blockchain network is not a 'fully' connected graph (see Fig. 1.7). In the remainder, we refer to the set of nodes to which a node is connected in a blockchain network as its 'peers'.

1.7.1 Public and private blockchain networks

One important way to characterize blockchain systems is whether they rely on a public or private blockchain network.

A public blockchain network is one in which anybody can become a user. In these blockchain networks, a consortium, a foundation, an organization or a group thereof maintains and updates when deemed necessary the specifications of the protocol used by the nodes of the network. The blockchain protocol in this case is usually made public, in the form of white papers and open-source standard client implementations. In this way, anybody can inspect the protocol and the software applications implementing it, if needed. Several implementations of the protocol can be available. These are software programs (written in different languages and for different platforms) that anybody can download and install on a computer to be able to run a node of the blockchain network. The Bitcoin network is probably the most prominent example of a public blockchain network. At https://bitcoin.org , several implementations of the software required to run a node of the

Bitcoin blockchain network (referred to perhaps improperly as the “wallet”) are available, e.g., for Android/Ios mobile phones and for Windows/Mac/Linux desktops. Anyone can download this software and run a node of the Bitcoin network. A node obviously has an initial balance equal to 0 bitcoins, but it can receive bitcoins from other nodes.

A public blockchain network should be carefully designed to ensure its scalability. If successful, in fact, the number of nodes in the network may rapidly increase. For instance, the Bitcoin network was initially constituted by a few nodes and now it entails millions of nodes across the entire world.

A private blockchain network is one in which the access to the network is vetted. There is a mechanism in place to grant the right to participate into the blockchain network to a new node. The rights to access the blockchain network are granted by a consortium, foundation, organization or group thereof who created the network in the first place. This type of blockchain systems normally are deployed in specific business scenarios. In such scenarios, a well-defined set of business partners, such as the organizations involved along a supply chain, decide to support at least part of their business using a blockchain system. First, they agree on the protocol used by the system. Then, they also decide which nodes and under which conditions can be admitted into the system. The blockchain protocol and its implementations, in this case, are likely to be private. In many cases, in fact, being admitted to the system may represent a source of competitive advantage for an organization. Therefore, the system users may have a strong incentive in maintaining the details regarding the system implementation, such as the protocol used, private. Because the access to the blockchain network is controlled, the nodes of a private blockchain network can be trusted to a higher degree than in a public blockchain network. This has important implications on the design of blockchain systems relying on a private blockchain network.

1.8 Data Management in a Blockchain Node: Transactions and Immutable Databases

The nodes of a blockchain system interact by exchanging digital messages. These messages are normally called transactions. A transaction is issued by one node, which we call the originator of the transaction. Depending on the application scenario, a transaction may contain different type of data and it may be addressed to one or more other nodes of a blockchain system. For instance, a Bitcoin transaction specifies a transfer of cryptocurrency tokens (bitcoins). This can be between two accounts (one sender and one recipient) or among multiple accounts (multiple senders and/or multiple recipients). In a blockchain system to monitor a supply chain, transactions may contain more complex data about the routing of shipping containers. These transactions may not be addressed by one user to one or more other specific users (as in the Bitcoin example), but simply issued by one user, such as a port authority, to be consulted by any other user of the system, e.g., shipping companies and custom offices.

Even if they signify a message intended to specific user(s), transactions in a blockchain system are usually broadcast to all the nodes of a blockchain. Technically, this broadcast is achieved using a so-called “gossip protocol”. The originator node sends a transaction that it wants to issue to its peers. These, in turn, broadcast the transaction to their peers and so on

until a transaction reaches all the nodes of a blockchain network. Here we see the connectivity property in blockchain network at work: a transaction can reach all nodes only if there is at least one path in the blockchain network to reach them. During the gossiping, transactions are also usually validated by each node that receives them. For instance, nodes check that transactions contain all the information required by the blockchain protocol. Transactions that fail this validation step are usually not forwarded by the nodes to other nodes. Fig. 1.8 shows an example of the gossiping protocol in action. The transaction T1 transfers 10 tokens from Alice to Luna. It has originated from Alice and it has reached Alice's peers (Bob, Carol, and Dave). Each node of the network maintains a "distributed immutable ledger" (we will explain this in detail later) with the balance of every node of the network. The transaction T1 has not reached Eva, yet. This is why, for Eva, the balance of Alice and Luna is still 35 and 2, respectively, while for all the other nodes that have received T1 the balance of Alice is 25 and Luna's is 12.

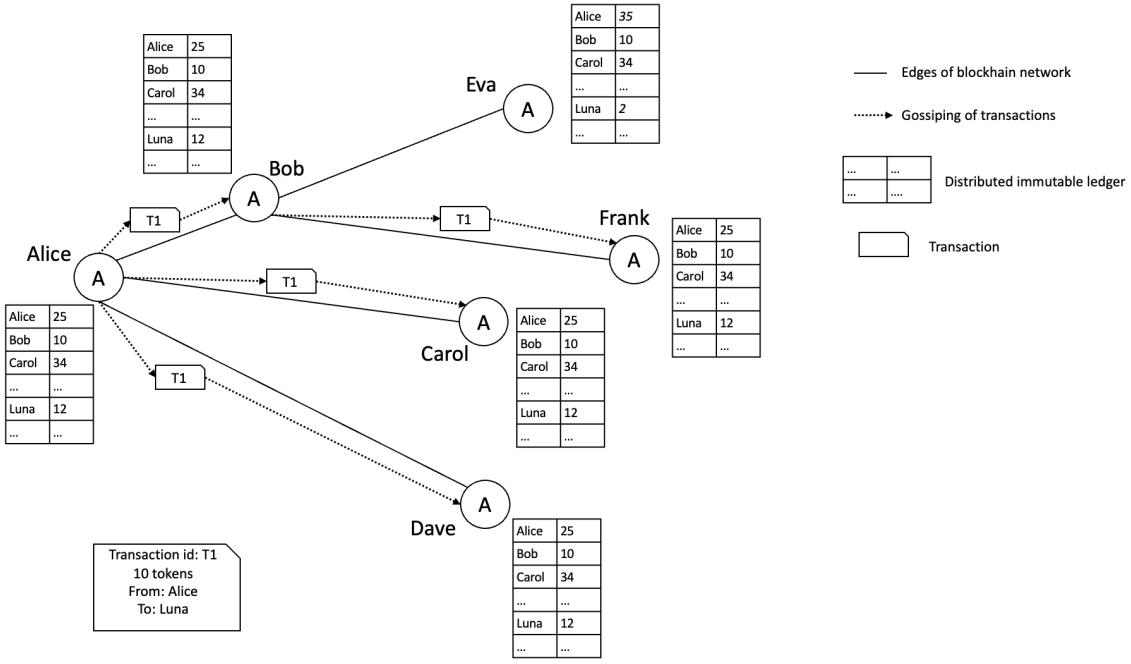


Fig. 1.8 – Gossiping of transactions and ledger update

Because nodes may be offline and/or may not implement all the functionality in a blockchain protocol, the gossiping of transactions can be unpredictable. This is normal in large-scale P2P networks, not only in blockchain networks. If one node issues a transaction A now and another node (or even the same node) issues a transaction B very close in time after A[2], we cannot assume that all the nodes in a blockchain network receive the transaction A before the transaction B. The order in which the transactions are received by a node depends on an extremely large number of variables, such as the paths of a blockchain network, the rules of the blockchain protocol, the availability of nodes, or the speed at

[2] What constitutes "close in time" in this example depends on the specific blockchain system under consideration. Transactions spaced even minutes in time may still be considered "close" in Bitcoin, whereas in other systems, such as Ethereum, transactions may be processed more quickly. More details are discussed for specific blockchain systems in Part 2.

which nodes validate transactions and forward them to their peers. This is an issue that we will revisit later in Section 4, where we discuss the need to introduce a consensus protocol to establish a proper order of the transactions agreed by all the nodes.

Based on what we have discussed thus far, it should be clear that a transaction in a blockchain system is not a direct private message issued by one node and addressed to one or more other nodes. We should instead view a transaction as a proposal to update the "state" of a blockchain system. If such a proposal is valid, then all the nodes in the blockchain network store it in their database to be able to reproduce the change of state entailed by it. For instance, the 'state' of a cryptocurrency system is captured by the balance of tokens of every node. Then, a cryptocurrency transaction in which node A transfers 1 bitcoin to node B can be seen as an update of the system state. Every node of the network should know eventually that the balance of node A must decrease by 1 token, whereas the balance of the node B must increase by 1 token. Hence, we can now see clearly that the ledger of the nodes in Fig. 1.8 shows the values of the variables describing the state of the system, i.e., the balance of every user. These variables are technically not immutable, since they are updated by every transaction. What is immutable is the sequence of the transactions that determines the value of the state variables (or, equivalently, the history of the values of the state variables).

As far as authentication is concerned, usually (ref. Chapter 2), a public key encryption mechanism is in place to guarantee the authentication of the transaction originator. Originators sign the transactions that they issue to a blockchain system using their private key. Any other node receiving a transaction verifies the identity of an originator using the originator's public key. This verification step is also part of the transaction validation. If the verification of the originator's identity fails, a node does not gossip a transaction to its peers.

Each node in a blockchain system stores the valid transactions that it receives in a database. As we discussed in Chapter 1, this database is immutable by design. This means that it is only possible to append new transactions to it, but it is not possible to modify, let alone delete, the transactions that are already stored in it.

1.8.1 Implementing data immutability by design

An issue that has remained open until now is understanding how it is possible to design a database that is immutable or, using the terminology introduced in the previous section, how to implement an immutable distributed ledger of transactions. This section solves this issue.

The fundamental technology behind the immutability of the ledger in a blockchain system is cryptographic hashing. In a nutshell, given a digital message (like a transaction in a blockchain system), cryptographic hashing allows to create a digest of it of fixed length that is practically unique.

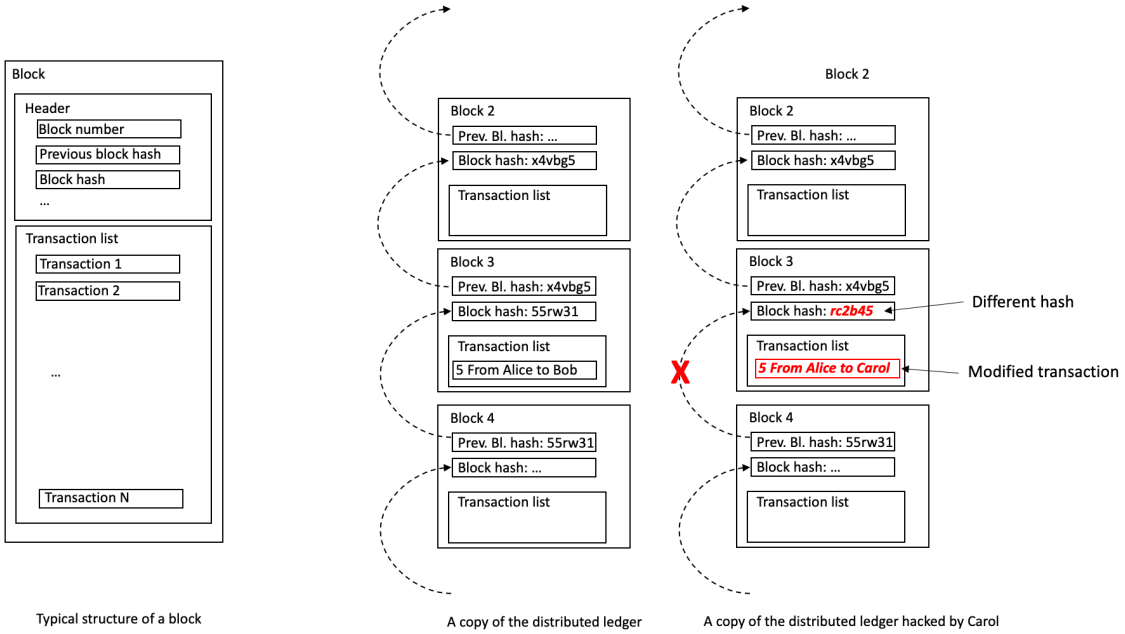


Fig. 1.8 – Block structure and block chaining to achieve immutability of the ledger

Before looking at how cryptographic hashing can be exploited for building an immutable ledger, let us define the concept of block in a ledger simply as the smallest amount of data that can be appended to it. A block normally contains (see Fig.1.8) a header and a list of transactions:

- The header of a block contains metadata that describe it, such as a unique identifier, a timestamp capturing the time instant at which it was created, or the number of transactions that it contains. Because blocks normally can be ordered in time using the timestamp, the unique identifier of a block is often a progressive number: block 1 precedes block 2, which precedes block 3 and so forth.
- The transaction list normally contains a list of valid transactions that have been issued (recently) by nodes of the blockchain systems. The number of transactions included in a block depends on the implementation. Normally, transactions are grouped into blocks for efficiency reasons. However, there are even blockchain systems where each block contains only a single transaction.

Immutability in a ledger is created by enforcing that a block contains, normally in its header, two additional fields:

- The hash of the block, i.e., a cryptographic hash of the content of the block (header and transaction list).
- The hash of the previous block. For instance, block 3 contains the hash of block 2 in its header, block 4 contains the hash of block 3 in its header, and so forth.

As shown in Fig1.8, the fact that a block contains the hash of its predecessor helps creating a ‘chain of blocks’ (sounds familiar?), in which consecutive blocks are linked by their matching hashes.

We are now ready to understand why creating a chain of blocks using the block hashes implicitly yields the immutability of the ledger storing the blocks. To do so, we should imagine what a malicious hacker (Carol in the example of Fig.1.8) should do in order to modify the content of the ledger:

- First, the hacker should modify the content of a block. Specifically, the hacker would aim at modifying the content of the transactions in a block. Fig 1.8 considers as an example a cryptocurrency system, where Carol wants to modify one transaction contained in block 3.
- Modifying the content of one block, however, changes the hash of that block. In fact, modifying even one single bit in a block changes its hash completely.
- Let us assume that the hacker succeeded somehow in modifying the content of one transaction in block 3. Now, the previous hash field in the header of block 4 no longer matches the hash of block 3. The hacker then must try to update the content of block 4, modifying the hash of the previous block to match the one of block 3, but ensuring that the hash of block 4 does not change. We know, however, that because of the property 1 of cryptographic hashing, this is (practically) impossible: if the content of block 4 changes, then its hash will also change.
- Alternatively, the hacker may try to modify the content of block 4 (for instance its header, which does not contain transactions) in the hope of making it such that the hash of block 4 stays the same. Because of property 2 of cryptographic hashing, however, we know that this is (practically) impossible, as well: it is impossible, in fact, that two 'versions' of the same block 4 map to the same hash value.
- Note that what we discussed so far applies when the hacker wants to modify the content of one specific copy of the distributed ledger. The same hacking process should also be applied to all the copies of the ledger, i.e., the copies held by every node participating in a blockchain system.

To summarize, the chain of blocks created by the chain of hashes in the header of the blocks ensures that a blockchain ledger cannot be modified. More precisely, it ensures that it is very easy for any node to quickly verify that its copy of the ledger has been modified and where (which block). This can be done simply by checking the chain of block hashes.

To conclude, there is one last (easy) problem that we need to solve when creating the chain of blocks, which concerns the start of such chain. In other words, what is the previous block of the first block of an immutable ledger? This problem is simply solved by assuming that every ledger in a blockchain system begins with a conventional "genesis block", for which the previous hash block is set to 0. Depending on the specific protocol implemented by a blockchain system, the genesis block may be very minimal, e.g., containing only a header, or may already contain a list of transactions.

1.9 Consensus Mechanism

The last technology contributing to the functioning of blockchain systems is the consensus mechanism. The consensus mechanism ensures that all the nodes in a blockchain system maintain consistently the same copy of the ledger. In other words, the consensus mechanism ensures that all the nodes agree on the content of the ledger, i.e., which blocks

and which transactions in them are included and in which order, at any point in time. This guarantees that, by replaying the transactions stored in the blocks of the ledger, every node can reconstruct the same current state of the blockchain system (or, at least, can verify that the current state that they are using is the correct one). For instance, in a cryptocurrency system, by replaying all the transactions from the creation of the system, any node can verify the correctness of the balance of every other node.

More in detail, the challenges of designing a consensus mechanism are better understood considering the following problem: how is it possible to establish the order of multiple transactions issued in a short time span? In the context of cryptocurrencies, this challenge goes under the label of preventing the "double spending" of tokens.

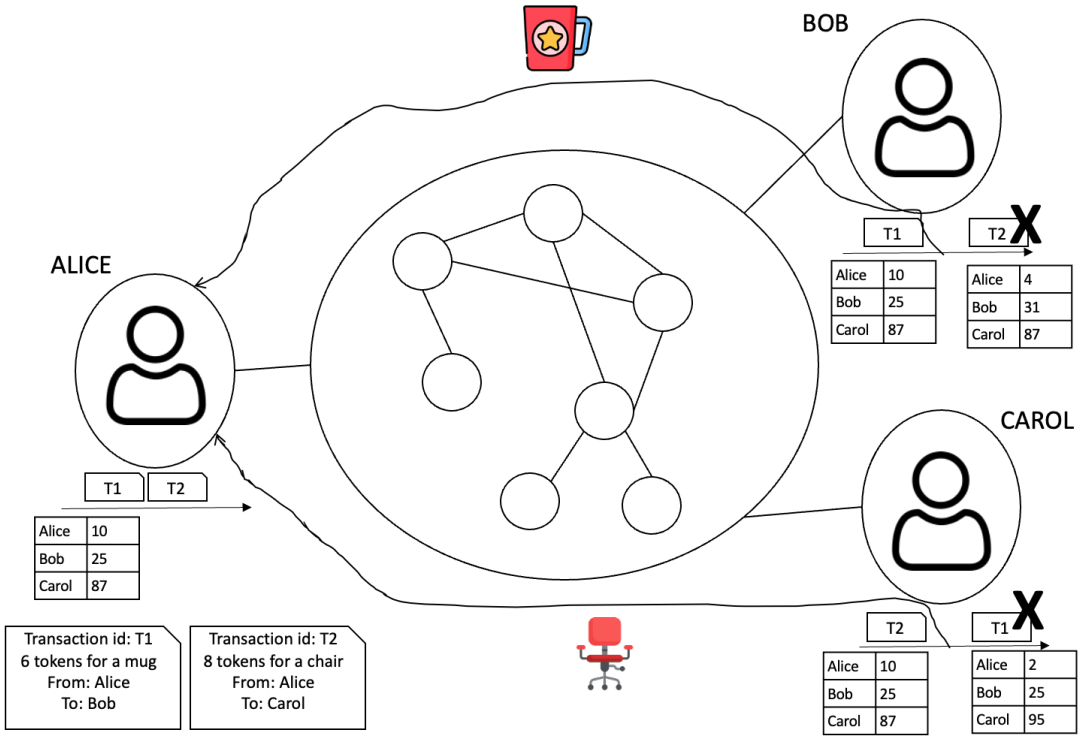


Fig. 1.9 – Double spending in blockchain exemplified

To introduce the challenge of double spending, we consider a simple blockchain system in which every transaction records a transfer of a certain number of tokens of a cryptocurrency between two users. Now (see Fig. 1.9), let us also assume that the user Alice has 10 CGM tokens and wants to use them to buy a mug from the user Bob for 6 CGM tokens, and a chair from the user Carol for 8 tokens. Obviously, Alice does not have enough tokens to pay for both items, but let us consider the following scenario:

- Alice creates one transaction (T1) transferring 6 tokens to Bob and one transaction (T2) transferring 8 tokens to Carol. Alice issues both transactions to the blockchain network in a short time span. Based on what discussed earlier in this chapter, we cannot guarantee that all other nodes receive these transactions (issued close in time) in the same order. Let us assume then that Bob receives T1 before T2, whereas Carol receives T2 before T1.

- Based on Alice's balance (10 tokens), any node considers the first transaction received between T1 and T2 valid, and rejects the second one as invalid (if T1 is processed first by a node, then for that node Alice has only 4 tokens left on her balance, which is not enough to process also T2; similarly for a node that processes T2 first).
- Now, if Bob receives T1 before T2, he thinks to have received the payment that he requested for the mug. Therefore, he sends the mug to Alice. Then, Bob rejects the transaction T2 because it is inconsistent with the current state of the blockchain system (i.e., Alice has only 4 tokens left). Similarly, if Carol receives T2 before T1, she thinks to have received the payment for the chair and sends it to Alice, rejecting the transaction T1.
- In the end, Alice may receive both the mug and the chair, somehow having managed to spend some of her tokens "twice" (i.e., double spending).

This simple example is archetypical of more complex issues that can arise in a blockchain system because of the ordering of transactions. In a blockchain system, we cannot rely on a central authority to fix this problem. If the exchange of tokens among Alice, Bob and Carol, for instance, is mediated by a bank, the problem of double spending of Fig. 1.9 would not exist: the bank would clear only the first transaction that either Bob or Carol originated, rejecting the second one because of insufficient funds in Alice's account. In a blockchain system, where only users exist and no central authority with "special powers" is allowed, the users are left alone to solve this problem. In practice, the double spending problem is solved devising rules to establish a consensus on which transactions are valid and in which order the transactions should be considered by nodes in the ledger.

The consensus mechanism is implementation-specific and, for understanding it, one should look in detail at the implementation and functioning of specific blockchain systems, like Bitcoin or Ethereum.

The contents of this document are extracted from:

Comuzzi, M., Grefen, P., & Meroni, G. (2023). *Blockchain for Business: IT Principles into Practice*. Routledge. [https://doi.org/10.4324/9781003321187]

Chapter 13

Project management

Giulio Nicelli

An introduction to IT Project Management

Project management is the discipline that deals with managing business projects, that is, sets of coordinated and controlled activities to achieve one or more predefined objectives. Within IT projects, project management has the role of planning, organizing, monitoring and closing the activities related to the development and maintenance of software systems.

A project can be traditionally defined as:

- A unique, transient endeavor, undertaken to achieve planned objectives, which could be defined in terms of outputs, outcomes or benefits. […] Time, cost and quality are the building blocks of every project. (Association for Project Management 2012)
- A set of people and other resources temporarily assembled to reach a specific objective, normally with a fixed time period (Graham 1989)
- A temporary endeavor undertaken to create a unique product or service (Project Management Institute 2013)

Uniqueness, limitation of time and resources can be chosen as key words in the definitions. The pillars of a project are unique scope (the goal to be achieved), time/duration, and resources, often over-simplified as costs (usually defined as the available budget). Project success has traditionally been measured by these three simple categories. The three dimensions are the classic dimensions of a project. Each project management decision can act on a specific dimension, but they cannot be managed as independent because they are intertwined in their definition. The trade-off between the dimensions must be considered because the time it takes to get to market, the unit cost incurred, and the scope (and thus quality), understood as product performance, are related to each other and this is reflected in the overall quality of the outcome.

The golden project triangle is the concept that refers to the trade-offs between cost, time and quality in project management1. Drury-Grogan (2014) applied this concept to agile software development teams and examined how their iteration objectives and critical decisions relate to these success factors. She identified four categories of decisions that affect the golden triangle: quality, dividing work, iteration amendments and team satisfaction.

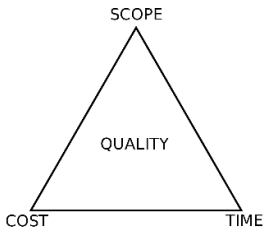


Figure 1: Project Management golden triangle

A description of the golden project triangle by Drury-Grogan (2014) is:

"The golden triangle of PM success factors has been used for decades as a way to measure PM success. The three sides of this triangle are cost (budget), time (schedule) and quality (scope). The logic is that if one side changes then another side must also change; for example, if a project's scope increases then either its budget or schedule must also increase."1

It is uncommon for the scope of a project to be entirely clear from the outset. In a conventional, established process, members of a group are familiar with the project's scope, including potential issues, budget

[1] https://www.sciencedirect.com/science/article/pii/S0950584913002176

constraints, and the expected time required to complete the work. However, when a project is undertaken by a startup or a large, established organization seeking to undertake a digital transformation, the scope may be less clear. While the initial steps may be understood, the project may require a departure from traditional practices, which can cause the scope to change incrementally over time. With limited resources and a restricted scope, the project may gradually expand and adapt to evolving needs.

Based on the environment in which a project is being implemented (as shown in Table 1) and the components of the project triangle that must be adhered to, three primary project management methodologies can be identified, as outlined in Table 1. A project management methodology is characterized as the collection of procedures, techniques, methods, regulations, templates, and best practices employed on a project (Project Management Institute 2013). These methodologies are as follows:

- Traditional Project Management
- Agile Project Management
- Extreme Project Management

		Scope	Time	Costs
	TPM	Fixed	Fixed	Fixed
	APM	Variable	Possibly fixed	Possibly fixed
	XPM	Not well defined	Re-Plannable	Re-Financeable

Table 1: Main categories of project management

The evolution of project management in IT projects has been influenced by the technological and organizational transformations that have characterized the IT sector from the 70s to today. In particular, some main phases can be identified:

- In the 70s-80s the waterfall model was used, based on a linear and rigid sequence of phases (analysis, design, coding, testing and release), with little interaction between the various stakeholders and low flexibility to changes.
- In the 90s iterative and incremental models were widespread, such as RUP (Rational Unified Process), which envisaged a cyclical and adaptive approach to the project, with greater participation of the customer and better management of requirements.
- From 2000 onwards agile methodologies have been established, such as Scrum and XP (Extreme Programming), which have introduced concepts such as collaborative development, frequent delivery of working products, continuous feedback and quick response to changes.

Dealing with uncertainty

Uncertainty in IT projects is the degree of unpredictability or variability of the project's inputs, processes and outputs. Uncertainty can arise from various sources, such as technology, requirements, human factors and environment. Uncertainty can affect the project's performance and outcome by increasing complexity, ambiguity and risk, but uncertainty can also create challenges and opportunities for learning and innovation.

Uncertainty in IT projects can have a significant impact on the correction costs during the project lifecycle. Correction costs are the costs incurred to fix errors or defects that are detected in the project deliverables or processes. Correction costs can be classified into two types: prevention costs and failure costs. Prevention costs are the costs spent to avoid errors or defects from occurring in the first place, such as quality planning, training, testing and inspection. Failure costs are the costs incurred to correct errors or defects after they have occurred, such as rework, debugging, warranty and litigation.

Uncertainty in IT projects can increase both prevention and failure costs, depending on the source and timing of uncertainty. For example, technology uncertainty can increase prevention costs if the project team needs to invest more time and resources to learn and master new or complex technologies. Technology uncertainty can also increase failure costs if the project team encounters technical problems or glitches that require rework or debugging. Similarly, requirements uncertainty can increase prevention costs if the project team needs to spend more effort and money to elicit and validate requirements from multiple stakeholders. Requirements uncertainty can also increase failure costs if the project team delivers a product or service that does not meet the expectations or needs of the customers or users.

Therefore, uncertainty in IT projects can have a negative impact on the project's budget and profitability by increasing correction costs during the project lifecycle. To mitigate this impact, IT projects need to adopt effective strategies for managing uncertainty and ensuring quality throughout the project phases.

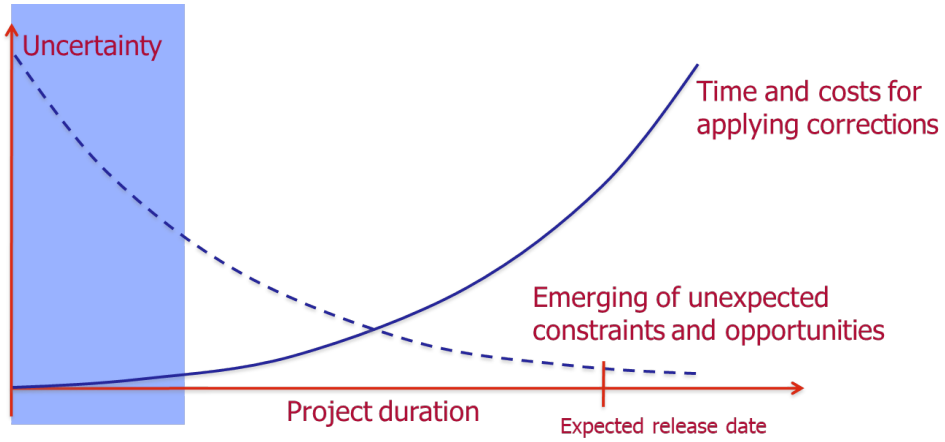


Figure 2: Uncertainty and correction costs during a project

Uncertainty in IT projects is a common challenge that can affect the project's performance and outcome. Uncertainty can arise from various sources, but two of the most important ones are technology and requirements. Technology uncertainty refers to the degree of novelty and complexity of the technology used or developed in the project, as well as the availability and reliability of the technical resources and skills. Requirements uncertainty refers to the degree of clarity and stability of the project's scope, objectives and specifications, as well as the involvement and feedback of the stakeholders. Both types of uncertainty can have significant impacts on the project's cost, time, quality and risk.

Technology uncertainty can pose difficulties for IT projects because it can increase the technical complexity and unpredictability of the project tasks, as well as create gaps between the expected and actual performance of the technology. Technology uncertainty can also affect the project team's ability to plan, coordinate and communicate effectively, as well as to cope with changes and problems that may arise during the project

lifecycle. To reduce technology uncertainty, IT projects need to adopt appropriate methods and tools for technology assessment, selection, integration and testing, as well as to allocate sufficient time and resources for technical learning and adaptation.

Requirements uncertainty can pose challenges for IT projects because it can increase the ambiguity and volatility of the project's scope, objectives and specifications, as well as create conflicts or mismatches between the expectations and needs of different stakeholders. Requirements uncertainty can also affect the project team's ability to define, prioritize and deliver value to the customers or users of the project outcome. To reduce requirements uncertainty, IT projects need to adopt suitable techniques for eliciting, analyzing, validating and managing requirements throughout the project lifecycle.

The Stacey Landscape Diagram (Figure 3) is a visual tool used in project management to help understand the complexity of a project or situation. The diagram was developed by Ralph Douglas Stacey and is often used in the context of organizational change and strategy implementation.

The Stacey Landscape Diagram consists of two axes, one horizontal and one vertical. The horizontal axis represents the level of certainty or agreement about the problem or situation at hand, ranging from highly certain to highly uncertain. The vertical axis represents the level of agreement or alignment among stakeholders involved in the project, ranging from highly aligned to highly conflicted.

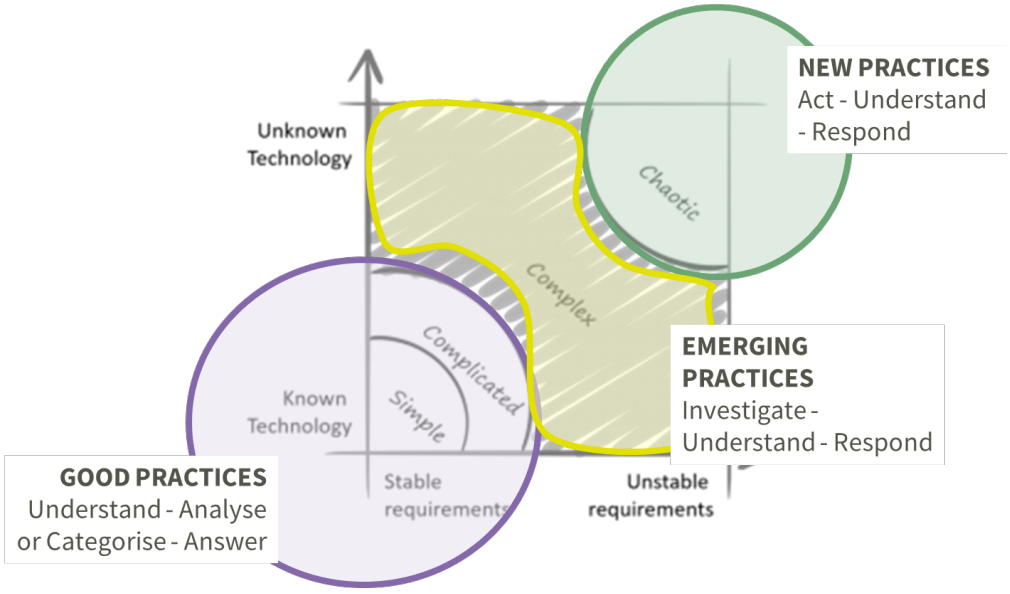


Figure 3: Stacey Landscape Diagram / Matrix

The resulting four quadrants of the diagram represent different project management approaches that may be appropriate for the given situation:

- Simple: In this quadrant, the problem or situation is highly certain, and stakeholders are highly aligned. This calls for a straightforward, directive approach to project management, where goals are clearly defined, and tasks are assigned in a top-down manner.

- Complicated: In this quadrant, the problem or situation is highly certain, but stakeholders may not be fully aligned. This calls for a more analytical, expert-driven approach, where project managers may need to rely on the expertise of specialized team members to solve complex problems.
- Complex: In this quadrant, the problem or situation is highly uncertain, but stakeholders are highly aligned. This calls for a more collaborative, exploratory approach to project management, where team members work together to experiment and adapt to changing circumstances.
- Chaotic: In this quadrant, the problem or situation is highly uncertain, and stakeholders are highly conflicted. This calls for a crisis-driven approach to project management, where urgent action is required to stabilize the situation and prevent further damage.

By using the Stacey Landscape Diagram, project managers can gain a better understanding of the complexity of a given situation and choose the most appropriate project management approach.

Traditional Project management

In a traditional project management setting, the approach is based on methods that can be applied uniformly across different projects. Standardization is believed to contribute to the robustness and flexibility of a wide range of projects. This traditional, rational, and normative approach views projects as relatively simple, predictable, and linear, with clearly defined boundaries. It assumes that similar projects have a similar structure, making it easy to outline a detailed plan to be followed with minimal.

The goal of this approach is to optimize and efficiently follow the initial predetermined project plan. However, while the robustness of the approach is often emphasized as one of its main advantages, it is increasingly mentioned as a disadvantage. The progressive evolution of business environments and projects, with their increasing number of tasks, complex relationships, and uncertainties, reflects the limitations of traditional project management, which heavily relies on structured hierarchies of linear tasks.

Another major flaw in traditional project management is the assumption that a project can be structured solely based on its expected outcomes, without considering its surrounding environment. It assumes that none of the three key components of a project (time, scope, and cost) have any uncertainty. However, it is highly unlikely that a project will flow unchanged, as external conditions or project parameters may change.

The factors for which a traditional project management approach is no longer recommended for a wide range of projects include structural complexity, uncertainty in goal definition, and project time constraints (Williams 2005).

This approach is commonly referred to as Waterfall or Stage Gate because each phase of the project must be fully completed before progressing to the next. Just as a natural waterfall flows from top to bottom without skipping any stages, in this approach, the product cannot be designed until the analysis phase is complete, and it cannot be tested until the implementation phase is finished. This is illustrated in Figure 4.

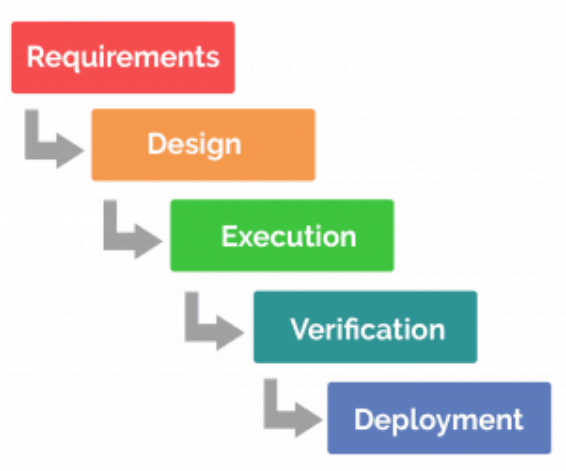


Figure 4: Waterfall approach

The Waterfall approach is a traditional and linear project management methodology consisting of sequential and distinct phases that must be completed before proceeding to the next. This approach is widely used in software development, construction, and other industries that require a highly structured process. The Waterfall model is designed to be a systematic and methodical approach that ensures project objectives are met on time, within budget, and with high quality.

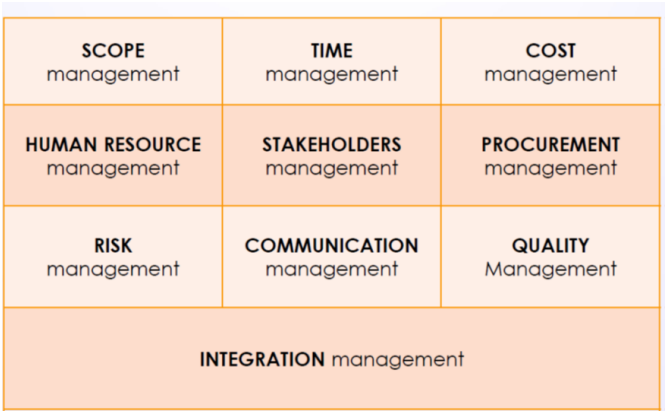
- The first phase of the Waterfall approach is the requirement gathering phase, where the project team identifies the requirements and objectives of the project. This includes defining the scope, goals, and deliverables of the project. In this phase, the project manager works with stakeholders to define the project requirements in detail. They document and prioritize the requirements and develop a requirements traceability matrix that will be used to track the requirements throughout the project.
- The second phase is the design phase, where the project team develops a detailed design specification based on the requirements gathered in the previous phase. The design specification outlines the technical details of the project, including architecture, algorithms, and data structures. This phase also includes creating mockups and prototypes to visualize the final product. The design phase is crucial in ensuring that the final product is both functional and meets the requirements of the stakeholders.
- The third phase is the implementation phase, where the design specifications are turned into actual code. The project team builds the system, and the developers write the software code and integrate the different components. This phase also includes conducting unit testing, which verifies that individual components of the system work correctly.
- The fourth phase is the testing phase, where the project team tests the system to ensure that it meets the requirements outlined in the first phase. Testing includes functional testing, which verifies that the system performs as intended, and integration testing, which verifies that the different components of the system work together correctly. This phase also includes user acceptance testing, where the stakeholders test the system to ensure that it meets their needs.
- The final phase of the Waterfall approach is the deployment phase, where the system is installed and made available to the end-users. This phase includes activities such as training, documentation, and support. The project team provides training to the end-users, creates user documentation, and provides ongoing support to ensure the system is stable and meets the needs of the stakeholders.

As seen, Waterfall approach is a highly structured and sequential approach to project management that emphasizes careful planning and execution. While it can be an effective methodology for projects that have well-defined requirements and a predictable outcome, it can be challenging to use for projects with uncertain requirements or in fast-paced environments. Nonetheless, the Waterfall approach remains a popular and

widely used project management methodology, especially in large-scale software development and engineering projects.

The PMBOK guidelines identify ten distinct knowledge areas or "souls" that are critical to successful project management:

- Project Scope Management: Defining, documenting, and controlling project scope.
- Project Time Management: Creating a project schedule that identifies and sequences all project activities.
- Project Cost Management: Developing a budget for the project and monitoring and controlling project costs.
- Project Quality Management: Ensuring project deliverables meet the quality requirements specified in the project scope.
- Project Human Resource Management: Identifying and managing all human resources and competences required for project completion.
- Project Communications Management: Managing project communications both internally within the project team and externally with stakeholders.
- Project Risk Management: Identifying and managing project risks throughout the project lifecycle.
- Project Procurement Management: Identifying and managing project procurement activities, including contracts and vendor relationships.
- Project Stakeholder Management: Identifying project stakeholders and managing their expectations and engagement throughout the project lifecycle.
- Project Integration Management: Coordinating all other knowledge areas throughout the project lifecycle.



Each of these knowledge areas is critical to the success of any project, and effective project managers must be knowledgeable and skilled in all of them. By understanding and applying the principles and practices of each knowledge area, project managers can ensure that projects are completed on time, within budget, and to the satisfaction of all stakeholders.

The following paragraphs will present some basic tools that can be used to address some of the key areas.

WBS- Work Breakdown Structure

The Work Breakdown Structure (WBS) is a foundational tool used in project management to break down complex projects into smaller, more manageable components. The WBS provides a hierarchical decomposition of the project scope, which is essential for effective project planning, budgeting, and scheduling. There are two main types of WBS: the activity-based WBS and the deliverable-based WBS. The activity-based WBS breaks down the project scope into smaller, discrete activities or tasks, while the deliverable-based WBS breaks down the project scope into tangible, measurable deliverables. Both approaches have their advantages and disadvantages, and the choice between them will depend on the specific needs of the project and the preferences of the project manager.

For example, an ERP (Enterprise Resource Planning) implementation project can be a complex and multifaceted undertaking that requires careful planning and execution. To effectively manage the project, a work breakdown structure (WBS) can be used to break down the project into smaller, more manageable components. One type of WBS is the activity based WBS, which is structured around the activities or tasks that need to be performed to complete the project.

In an activity based WBS for an ERP implementation project, the project is broken down into a series of tasks or activities that need to be performed in order to successfully implement the system. This could include tasks such as requirements gathering, system design, data conversion, testing, and training.

For example, in the requirements gathering phase, the project team would work with the key stakeholders to identify the specific needs and requirements of the organization. This could include gathering information about existing business processes, data structures, and reporting requirements. This phase could be further broken down into activities such as conducting interviews, reviewing documentation, and analyzing data.

Once the requirements have been gathered, the next phase of the project could be system design. This would involve designing the system to meet the specific needs and requirements of the organization. Activities in this phase could include designing workflows, configuring the system, and developing customizations or integrations.

Another key activity in an ERP implementation project is data conversion. This involves migrating data from the existing systems into the new ERP system. This could include activities such as data mapping, data cleansing, and data validation.

Testing is another critical activity in an ERP implementation project. This involves testing the system to ensure that it is functioning as expected and meets the requirements of the organization. Activities in this phase could include developing test cases, conducting system testing, and performing user acceptance testing.

Finally, training is a critical activity in an ERP implementation project. This involves training end-users on how to use the new system effectively. Activities in this phase could include developing training materials, conducting training sessions, and providing ongoing support to end-users.

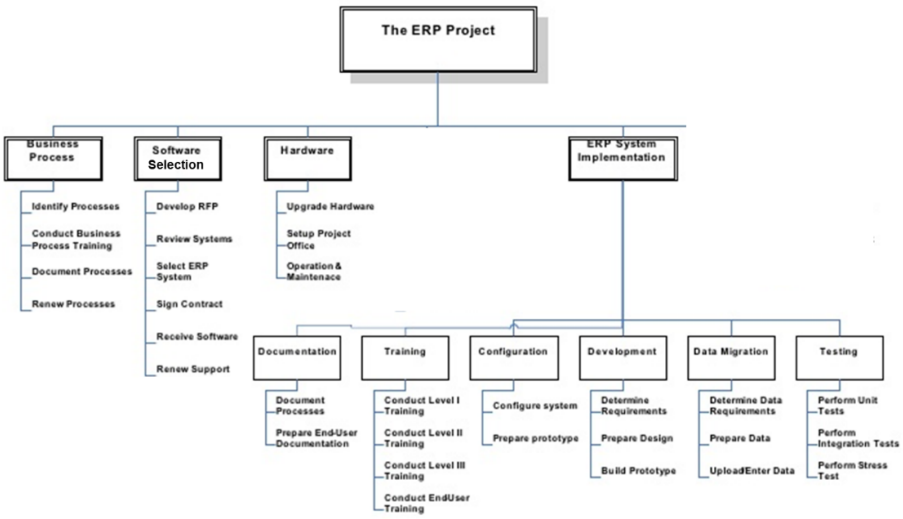


Figure 5: Example of an activity based WBS for an ERP Project

The deliverable-based Work Breakdown Structure (WBS), instead, organizes the project work into deliverables, which are the tangible outcomes or results of the project. In other words, the deliverables are the products, services, or results that are created, developed, or produced during the project. The deliverable based WBS defines the project scope in terms of the final products or outcomes that are expected from the project, and then breaks down each deliverable into smaller, manageable pieces of work.

In an ERP implementation project, the deliverables may include software modules, data migration plans, training programs, user manuals, and system documentation. For example, the ERP software modules can be broken down into specific deliverables such as inventory management, order management, financial accounting, and human resources management. Each of these deliverables can be further broken down into smaller components, such as user interface design, system integration testing, and data mapping.

The main advantage of the deliverable based WBS is that it provides a clear and structured approach to managing the project scope, as it focuses on the final outcomes or results of the project rather than the activities that need to be performed. By identifying and breaking down the deliverables into smaller, manageable pieces of work, project managers can better plan, organize, and control the project activities, resources, and schedules. Moreover, the deliverable based WBS facilitates communication and collaboration among project team members, stakeholders, and customers, as it provides a common language and understanding of the project scope and objectives.

However, one of the challenges of using the deliverable based WBS is that it requires a high level of expertise and knowledge of the project domain.

Gantt Chart

A Gantt chart is a popular tool used in project management to visually represent a project's schedule, tasks, and timeline. It was invented by Henry Gantt in the early 1900s, and it remains a widely used project management tool today.

A Gantt chart consists of a horizontal timeline and a series of bars or rectangles that represent each task in the project. The bars are arranged in order of the tasks' start and end dates, and they are color-coded to indicate the task's status (e.g., completed, in progress, not started). The chart also includes labels, such as task names and deadlines, to provide additional information about each task.

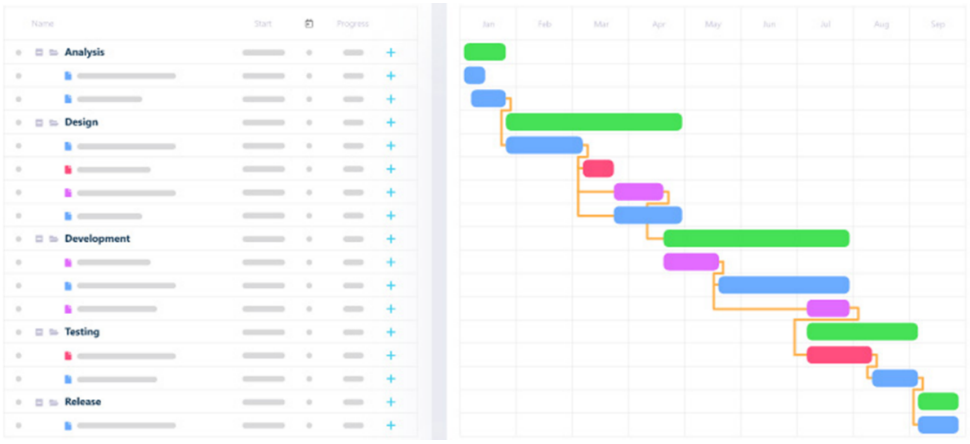


Figure 6: Gantt Chart example for a software development project

In IT project management, Gantt charts are used to plan and manage the software development process. They help teams to track project tasks, dependencies, milestones, and deadlines, and can be used to manage project resources such as personnel, equipment, and budgets.

One of the main advantages of using a Gantt chart in IT project management is that it provides a clear overview of the project's timeline and progress. This allows project managers to quickly identify any delays or potential roadblocks and adjust the project plan accordingly. It also helps team members to see their individual tasks and how they fit into the overall project timeline, which can increase motivation and productivity.

Gantt charts are also useful for managing project dependencies. By clearly showing the order in which tasks need to be completed and the dependencies between them, the chart helps to ensure that the project is completed on time and on budget. This can be especially important in IT project management, where complex software development projects often have multiple interdependent tasks that need to be completed in a specific order.

CPM – Critical Path Method

The Critical Path Method (CPM) is a project management technique used to identify the longest path of tasks in a project schedule, which determines the project's critical path. This method allows project managers to identify the most critical tasks and prioritize them to complete the project on time. The steps that should be performed to use this method during an IT project planning phase are the following:

1. Identify the Project Tasks: First, identify all the tasks required to complete the project. Create a task/activity list, including start and end dates, dependencies, and resources required.
2. Determine the Task Durations: Next, estimate the duration of each task. This can be done by consulting with subject matter experts, reviewing historical data, or using project management software.
3. Determine Task Dependencies: Identify the dependencies between the tasks. Some tasks may be dependent on the completion of others, while others may be performed in parallel. It is essential to establish these dependencies accurately to determine the project's critical path.
4. Construct the Network Diagram: Create a network diagram (*Figure 7*) that represents the project tasks and their dependencies. This can be done using specialized software or by drawing the diagram manually.

5. Identify the Critical Path: Calculate the duration of each path in the network diagram. The critical path is the longest path in the network diagram and represents the minimum amount of time required to complete the project.
6. Determine the Slack Time: Once the critical path has been identified, determine the slack time for each task. Slack time represents the amount of time a task can be delayed without delaying the project's overall completion time. To determine the slack time a forward scheduling followed by a backword scheduling should be performed and the difference between the latest possible planning and the earliest possible planning is the slack time of that activity. All the activities without slack time are the ones on the critical path.
7. Optimize the Schedule: Once the critical path and slack time have been identified, adjust the project schedule to optimize its timeline. This can be done by adjusting the sequence of tasks or allocating additional resources to critical tasks.

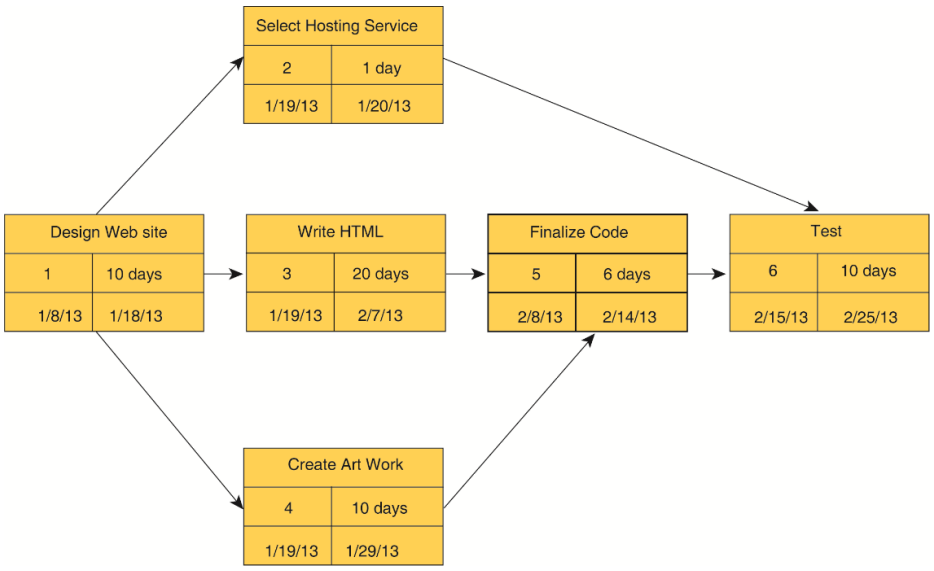


Figure 7: Simple network diagram that can be used to understand which is the critical path

Cost estimation techniques

Cost estimation is a critical component of IT project management. It involves predicting the costs associated with a project, including hardware, software, personnel, and other resources. The accuracy of cost estimation is essential for project planning, budgeting, and resource allocation. There are several cost estimation techniques used in IT project management, each with its strengths and weaknesses.

One commonly used cost estimation technique is the Analogous Estimation technique (also known as top-down estimation). It involves estimating the cost of a new project based on the cost of a similar project that was previously completed. This technique is useful when there is limited data available about the new project or when there is a tight deadline. However, the accuracy of this technique depends on the similarity between the new project and the previous project.

Another cost estimation technique is the Bottom-up Estimation technique. This technique involves estimating the cost of each project component, such as hardware, software, and personnel, and then adding them up to determine the overall project cost. This technique is useful when there is a detailed project plan available,

and when the project is complex and has many components. However, it can be time-consuming and may require extensive expertise in estimating the cost of each component.

Parametric Estimation technique is a cost estimation technique that uses statistical data and mathematical models to estimate project costs. This technique is useful when there is a lot of data available about similar projects, and when there is a high degree of certainty about the project's requirements. However, this technique requires a significant amount of data, and the accuracy of the estimates depends on the quality of the data and the model used.

The Delphi Estimation technique is another cost estimation technique that can be used in IT project management. This technique involves gathering opinions and estimates from a group of experts and then synthesizing them to arrive at a consensus estimate. This technique is useful when there is uncertainty about the project or when there are multiple factors that could impact the cost of the project. However, it can be time-consuming, and it may be difficult to get a consensus estimate if the experts have different opinions.

In general, a combination of several techniques can be used to arrive at a more accurate estimate. For example, it may be used the Analogous Estimation technique to get a rough estimate and then use the Bottom-up Estimation technique to refine the estimate based on the project's detailed plan. There are several cost estimation techniques used in IT project management, each with its strengths and weaknesses. The choice of technique depends on the project's complexity, the availability of data, and the level of uncertainty about the project.

Regardless of the technique used, cost estimation is not an exact science, and there are many factors that can impact the accuracy of the estimate. These include changes in project requirements, unexpected problems, and fluctuations in the cost of resources. Project managers must be prepared to adjust their estimates as the project progresses and to communicate any changes to stakeholders.

In IT project management, there are three types of costs that need to be considered: internal costs, external costs, and hidden costs.

- Internal costs are a critical component of project budgeting, as they represent the expenses associated with internal resources such as employees and infrastructure. These costs typically include salaries, benefits, and other costs associated with personnel such as training and development. In addition, internal costs can include costs associated with infrastructure such as servers, hardware, and software licenses.
- External costs represent the expenses associated with external resources such as vendors, contractors, and consultants. These costs can vary significantly based on the size and scope of the project and the resources required. For example, a software development project may require the services of an external software developer, while a hardware implementation project may require the services of a vendor to provide hardware and installation services.
- Hidden costs are often overlooked in project budgeting but can have a significant impact on the project's financial success. Hidden costs can include expenses such as equipment repairs or replacement, lost productivity due to project delays, and additional training costs associated with new systems or processes. For example, a hidden cost of an IT project might be the additional time required for employees to learn a new system or process. This can result in lost productivity and potentially increased labour costs as employees adjust to the new system.

It is important for IT project managers to consider all three types of costs when developing a project budget. By identifying and accounting for all costs associated with the project, project managers can ensure that the project is financially feasible and avoid unexpected expenses that may arise later in the project lifecycle. Effective cost management is critical to the success of any IT project, and by considering internal costs,

external costs, and hidden costs, project managers can ensure that their projects are completed on time, within budget, and with the desired outcomes.

OBS - Organizational Breakdown Structure

An OBS is a hierarchical representation of the project's organizational structure, which provides a clear picture of the roles and responsibilities of project team members. It helps to define the project's boundaries, identify potential areas of conflict, and ensure effective communication and collaboration among team members. By using an OBS, project managers can better estimate resource requirements and reduce the risk of miscommunication.

An OBS typically starts at the top level with the project manager and includes other key roles such as the project sponsor, stakeholders, and team members. As the OBS is developed, it is broken down into sub-levels, with each level representing a functional area, department, or team within the organization.

Each level of the OBS includes specific roles and responsibilities, which help to define the project's scope and requirements. For example, the top-level of the OBS may include the project sponsor and project manager. The next level may include functional areas such as finance, marketing, and IT. The IT functional area may then be broken down further to include specific teams such as application development, infrastructure, and testing.

An OBS can be used to identify potential gaps or overlaps in the project team's responsibilities, ensuring that each team member has a clear understanding of their role in the project. It also helps to define the project's boundaries and identify potential areas of conflict.

Another benefit of an OBS is that it provides a clear structure for communication and collaboration among project team members. It can be used to ensure that everyone involved in the project has access to the information they need, which can help to reduce the risk of miscommunication and misunderstandings.

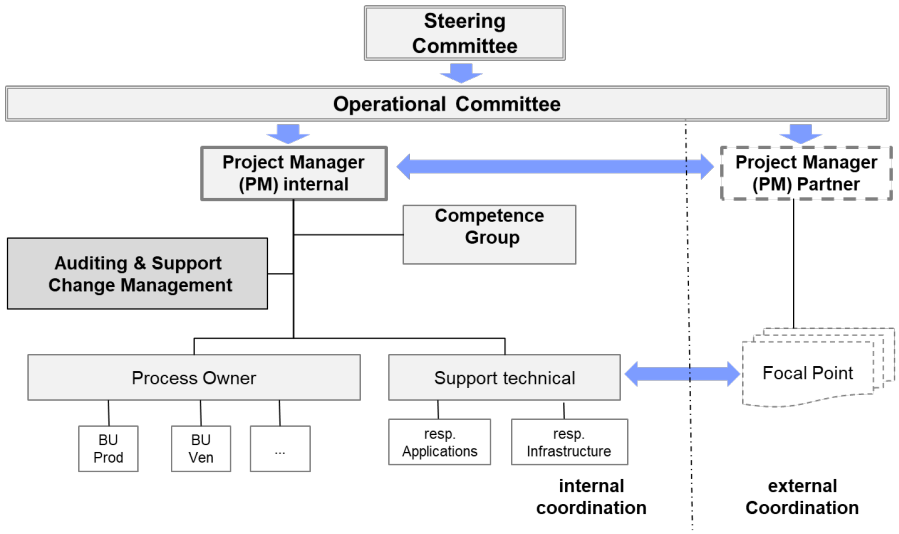


Figure 8: Example of an OBS for an ERP implementation project

In addition to defining roles and responsibilities, an OBS can also be used to identify resource requirements. By identifying the specific teams and departments involved in the project, project managers can better estimate the resources required for each stage of the project, including personnel, equipment, and materials.

While an OBS can be a valuable tool for IT project management, it is important to remember that it is not a static document. As the project evolves, the OBS may need to be revised to reflect changes in the project's

goals, scope, or requirements. It is important for project managers to regularly review and update the OBS to ensure that it remains relevant and useful throughout the project's lifecycle.

RACI Matrix

By cross-referencing wbs and obs, it is possible to create a useful matrix, the RACI Matrix, to clearly identify the responsibilities of different roles within the project.

The RACI matrix is a tool commonly used in IT project management to define roles and responsibilities for project team members. RACI stands for Responsible, Accountable, Consulted, and Informed. By using a RACI matrix, project managers can ensure that every task has a clear owner, and that all team members have a clear understanding of their role in the project. This helps to prevent confusion and ambiguity, leading to more efficient project completion and better outcomes.

The RACI matrix is typically represented in a table format, with each task or deliverable listed on the left-hand side of the table and each team member listed across the top. For each task or deliverable, team members are assigned a RACI code based on their level of involvement in the task:

- Responsible: The "R" in RACI stands for Responsible. This role is assigned to the team member who is responsible for completing the task or deliverable. This person is the one who actually does the work to ensure that the task is completed on time and to the required standard. They may work with other team members to complete the task, but they are ultimately responsible for the end result.
- Accountable: The "A" in RACI stands for Accountable. This role is assigned to the team member who is ultimately accountable for the success of the task, has the authority and accountability for the task or activity, and who delegates the work to the responsible person. This person is responsible for ensuring that the task is completed to the required standard, and that all stakeholders are informed of its progress. They have the final say on whether the task is complete or not, and they are responsible for the overall success or failure of the project. Often the accountability of every task in a project is assigned to the Project Manager.
- Consulted: The "C" in RACI stands for Consulted. This role is assigned to the team members who provide input and feedback on the task or deliverable. These team members are consulted on the task's progress and provide feedback and advice to the responsible team member. They may provide technical expertise or other support to help the responsible team member complete the task successfully.
- Informed: The "I" in RACI stands for Informed. This role is assigned to the team members who need to be informed of the task's progress but do not have a direct involvement in completing the task. These team members may be stakeholders in the project, such as senior management or customers, who need to be kept up to date on the project's progress. They do not have any responsibility for completing the task, but they need to be kept informed so they can make informed decisions about the project.

Step	Project Initiation	Project Executive	Project Manager	Business Analyst	Technical Architect	Application Developers
1	Task 1	C	A/R	C	I	I
2	Task 2	A	I	R	C	I
3	Task 3	A	I	R	C	I
4	Task 4	C	A	I	R	I

CIO/IDG

Figure 9: Example of a RACI Matrix

By using a RACI matrix, project managers can ensure that every task has a clear owner and that all team members have a clear understanding of their role in the project. This helps to prevent confusion and ambiguity regarding roles and responsibilities, which can lead to missed deadlines and miscommunication.

For example, in an IT project that involves the development of a new software application, a RACI matrix could be used to define the roles and responsibilities of the project team. The project manager may be accountable for the overall success of the project, while the software development team may be responsible for completing the development tasks. The quality assurance team may be consulted on the progress of the software development and provide feedback and suggestions for improvement. Finally, stakeholders such as the business owners and end-users may need to be informed of the software's progress but may not be directly involved in the development process.

Resource histogram

The resource histogram is a tool used to visualize the allocation of resources over the course of a project. It is a bar chart that displays the number of resources required for each task or activity in the project, as well as the time period in which those resources will be needed.

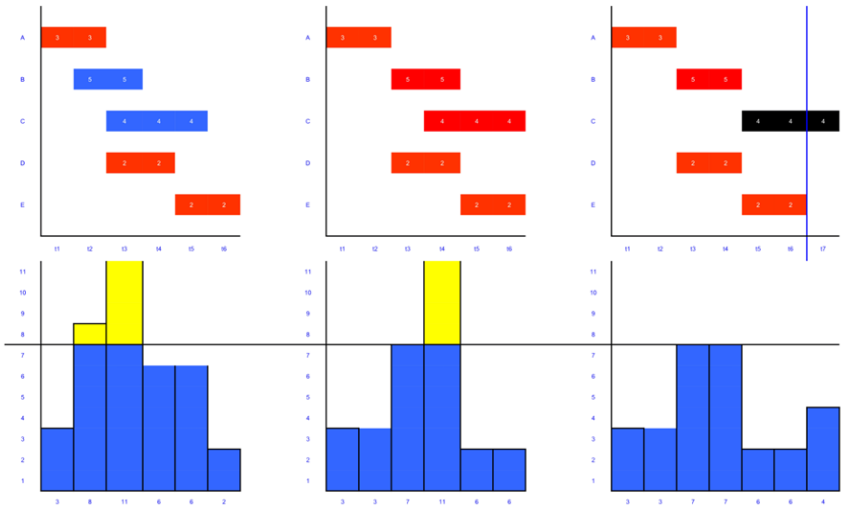


Figure 10: Example of a resource histogram (red activities are the ones on the critical path)

The resource histogram helps project managers to identify potential resource constraints and to make informed decisions about how to allocate resources effectively. It provides a visual representation of the availability of resources at any given time, allowing project managers to adjust their plans and schedules accordingly.

The resource histogram can be used to identify periods of peak demand for resources, such as when multiple tasks or activities require a significant amount of effort from the same team members or when there is a shortage of specific skills or expertise. This information can be used to adjust the project plan to ensure that resources are allocated in the most effective and efficient way possible.

For example, if the resource histogram shows that a particular team member is required to work on several critical tasks simultaneously, the project manager may need to either adjust the schedule to spread out the workload or find additional resources to help complete the tasks on time. Similarly, if the resource histogram shows that there is a shortage of a particular skill set or expertise, the project manager may need to either adjust the project plan to reduce the demand for that resource or bring in additional resources with the required skills.

If overallocation occurs, there are several possible approaches to rescheduling:

- Rescheduling can be done while considering fixed time constraints, such as a set deadline. This can be achieved by using the float margins of non-critical activities.
- Rescheduling can also be done while considering fixed time constraints and overbudget issues related to overtime and/or the need for additional resources.
- Alternatively, rescheduling can be done while taking into account limited resources, which may require adjustments to the project deadline.
- Another option is to de-scope certain aspects of the project, removing less critical tasks or reducing the overall scope in order to better allocate resources and meet project objectives.

In the left part of the example above (*Figure 10*) the histogram shows that with the first scheduling there is an overallocation in t2 an t3 because the number of required resources (i.e. SMD Standard Man Days) is higher than the ones that are available. Appling a rescheduling on both activity B and C the situation improves, but still there is an under-capacity issue in t4. In the right part of the figure further rescheduling was done by considering the usable resources fixed and thus leading to a period t delay due to the C activity going beyond the initially scheduled deadline.

The resource histogram is also useful for monitoring resource utilization over time. By comparing the actual resource usage to the planned resource allocation, project managers can identify potential issues early and take corrective action as needed. For example, if the resource histogram shows that a particular team member is consistently overworked or underutilized, the project manager may need to adjust the project plan to better balance the workload or provide additional training to help the team member develop new skills.

Stakeholder management: Mendelow’s Matrix

Stakeholder management is crucial in IT project management because it helps to ensure that everyone involved in the project is on the same page and working towards the same goal. Stakeholders can include project sponsors, end-users, developers, vendors, and other individuals or groups who have an interest in the project's success. Effective stakeholder management involves identifying and understanding each stakeholder's needs, expectations, and concerns, and then developing strategies to address them. By doing so, project managers can gain the support and buy-in of key stakeholders, which is essential for successful project implementation. In addition, stakeholder management helps to identify potential risks and issues early in the project, allowing project managers to mitigate them before they become major problems.

Mendelow's matrix (*Figure 11*) is a tool used in stakeholder management to identify the power and interest of stakeholders in a project. It provides a framework for understanding stakeholder needs and creating

strategies to manage them effectively. The matrix has four quadrants based on the stakeholders' power and interest in the project, and different strategies are recommended for each quadrant:

- High power, high interest: These stakeholders are the most important and should be managed closely. They have a significant impact on the project's success and can be either supporters or opponents. To manage them effectively, project managers should engage with them regularly, keep them informed, and seek their input and feedback.
- High power, low interest: These stakeholders could influence the project but may not be actively engaged. Project managers should keep them informed about the project's progress but not over-communicate. It is essential to address their concerns promptly and avoid any negative impact on the project.
- Low power, high interest: These stakeholders may not have the power to influence the project, but they are interested in its success. Project managers should keep them informed and involved in the project where possible. Their input can provide valuable insights and perspectives on the project's impact.
- Low power, low interest: These stakeholders have little influence or interest in the project. Project managers should not allocate significant resources to managing them but still keep them informed about the project's progress.

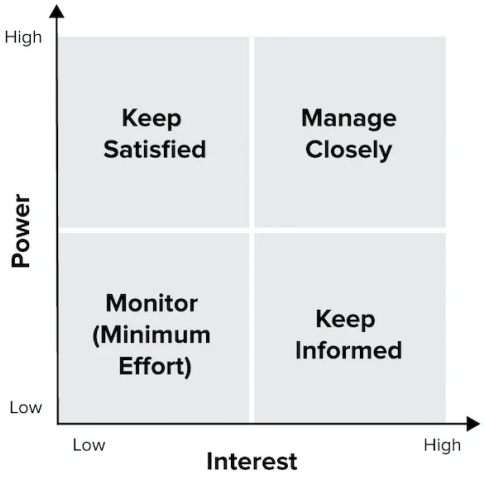


Figure 11: Mendelow's Matrix

Suppose a software development company is working on a new project to create a mobile application for a transportation company. The stakeholders in this project could include:

- High power, high interest: The CEO of the transportation company who has significant influence over the project's success.
- High power, low interest: A government regulator who has the power to enforce regulations related to transportation, but may not be actively engaged in the project.
- Low power, high interest: A group of users who are interested in the new mobile application's features and functionality.
- Low power, low interest: A small vendor who provides non-critical services to the transportation company and has no significant impact on the project's success.

Based on Mendelow's matrix, the project manager would prioritize the CEO of the transportation company as a key stakeholder and ensure that they are regularly updated on the project's progress, involved in key decisions, and kept satisfied with the project's outcomes. For the government regulator, the project manager would keep them informed about the project's compliance with regulations and address any concerns they may have promptly. For the group of users, the project manager would involve them in user testing and design decisions to ensure the mobile application meets their needs. The vendor would be informed about the project's progress, but not allocated significant resources to managing their involvement.

Project Risk Management

Risk management is an essential aspect of traditional IT project management, as it helps to identify, assess, and mitigate potential risks that can impact project outcomes. In traditional IT project management, risk management involves several steps, such as risk identification, risk assessment, risk prioritization, risk response planning, and risk monitoring and control.

Risk Identification

Risk identification is the first and crucial step in traditional IT project management's risk management process. It involves identifying potential risks that can affect project objectives, timelines, and budgets. Effective risk identification helps project teams to anticipate potential problems and take proactive measures to mitigate them before they occur.

There are different techniques for identifying risks in traditional IT project management, including brainstorming sessions, checklists, interviews, and historical data analysis. Brainstorming sessions are a common technique for identifying risks, where the project team gathers to identify potential risks through free-thinking and creative discussion. Checklists, on the other hand, are a useful tool for ensuring comprehensive risk identification. They provide a structured approach to identify potential risks, categorizing them into different types and areas, such as technical, environmental, financial, and human resources.

An Ishikawa diagram, also known as a fishbone diagram or cause-and-effect diagram, can be a valuable tool for IT project managers when it comes to identifying potential risks. To use an Ishikawa diagram, project managers begin by identifying the primary risk they wish to address, such as a potential delay in project delivery. They then work to identify all possible causes of this risk, such as inadequate resources, unclear project requirements, or lack of communication among team members. Once all potential causes have been identified, project managers can analyse them and determine which ones are most likely to cause the risk. From there, they can develop strategies to mitigate or eliminate those causes, thereby minimizing the likelihood of the risk occurring and increasing the likelihood of project success.

Interviews with stakeholders, subject matter experts, and project team members can also help to identify potential risks. By engaging these individuals, the project team can gain valuable insights into potential risks that they may not have considered. Historical data analysis is another technique for identifying risks. By reviewing data from similar projects, the project team can identify potential risks that have occurred in the past and prepare for them in the current project.

It is essential to involve the project team and stakeholders in risk identification to ensure that potential risks are identified from all angles. Project managers should create an environment that encourages open communication, where team members feel free to share their concerns and ideas about potential risks.

Risk Assessment

Risk assessment typically starts at the beginning of the project, during the planning phase, where the project team identifies potential risks based on their experience and knowledge. Risks can come in many forms, such as technical risks, resource risks, schedule risks, and financial risks. The project team should assess the likelihood and impact of each identified risk and determine the appropriate response strategy. There are several methods that can be used to assess the likelihood and impact of a risk, including qualitative and quantitative techniques.

Qualitative risk assessment involves assessing the likelihood and impact of a risk based on the experience and judgment of the project team. This method typically involves using a risk matrix (*Figure 12*), which is a tool that helps to visualize the likelihood and impact of a risk. The project team assigns a score to the likelihood and impact of the risk on a predefined scale, and the scores are then used to plot the risk on the risk matrix. Probability indicates the likelihood of an event occurring, while impact indicates the magnitude of the impact of a specific event. The risk matrix can be customized to reflect the specific needs of the project, in terms of number of levels and risk areas.

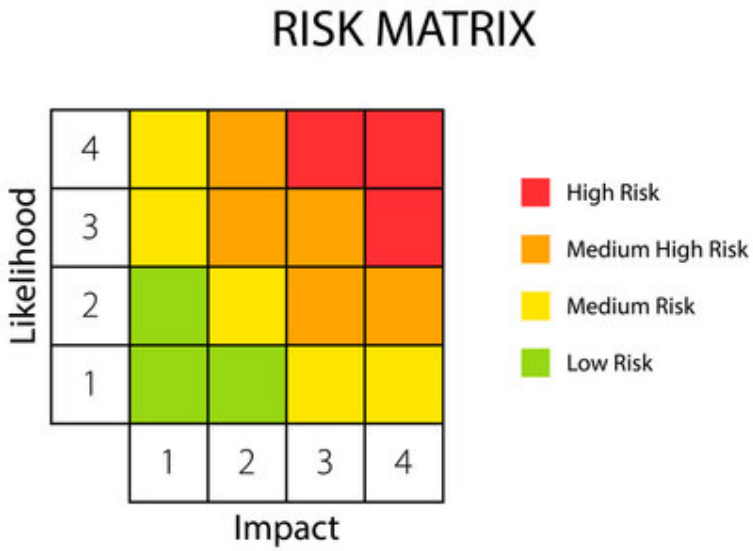


Figure 12: Example of a 4x4 Risk Matrix

It's important to note that the likelihood and impact of a risk are not always independent of each other. For example, a risk with a low likelihood but a high impact may be more significant than a risk with a high likelihood but a low impact. Therefore, the project team should consider both the likelihood and impact of a risk when assessing its significance.

Quantitative risk assessment involves using statistical analysis and modelling techniques to assess the likelihood and impact of a risk. This method typically requires more data and resources than qualitative risk assessment, but it can provide a more accurate and precise assessment of the risk. The project team can use techniques such as Monte Carlo simulation to model the impact of different scenarios on the project outcome. This can help to identify which risks are most likely to have a significant impact on the project and prioritize risk management efforts accordingly.

Risk Management

A risk management strategy involves developing a plan to minimize the likelihood and impact of risks throughout the project lifecycle. There are several risk management strategies that IT project managers can use to mitigate risk and ensure project success. These strategies include:

- Risk Mitigation: This strategy involves developing a plan to reduce the likelihood (prevention) or the impact (protection) of a risk if it occurs. For example, the project team may develop a disaster recovery plan to minimize the impact of a natural disaster on the project.
- Risk Avoidance: This strategy involves avoiding activities or events that could potentially result in risk. For example, if a particular technology is known to be unreliable, the project team may avoid using that technology to reduce the risk of project failure.
- Risk Transfer: This strategy involves transferring the risk to another party, such as an insurance company or a subcontractor. For example, the project team may transfer the risk of a data breach to a third-party cybersecurity provider.
- Risk Reduction: This strategy involves taking steps to reduce the likelihood and impact of a risk. For example, the project team may implement additional security measures to reduce the risk of a data breach.
- Risk Acceptance: This strategy involves accepting the risk and developing a plan to manage the potential impact. For example, the project team may accept the risk of a delay in the project schedule and develop a contingency plan to manage the impact of the delay.

A useful tool that can be used to manage the project risks is the Risk Plan Matrix (*Figure 13*) in which it can be found the list of the identified risks and all the related information (likelihood, impact, overall risk level, countermeasure, countermeasure cost, …), so it is a comprehensive matrix that outlines the potential risks to the project, their probability, impact, and the action plan to mitigate the risk.

The risk response planning section of the matrix involves developing an action plan to mitigate the risks. The plan should include the specific actions to be taken, the person/role responsible for implementing the plan, and the timeframe for implementation. The project team should prioritize the risks based on their impact and probability and focus on the high-risk areas first.

CAUSE	EFFECT	PROBABILITY '(START)	IMPACT (START)	RISK LEVEL (START)	COUNTERMEASURE	COST (€)	OWNER	PROBABILITY '(END)	IMPACT (END)	RISK LEVEL (END)	NOTES
Technolog provider Failure	Stop of production systems	5	5	25	Change technology platform	€ 400k	Board	5	0	0	
Bugs development	application Crash	4	3	12	Increase test coverage	€ 20k	Dir. IT	2	3	6	
…	…	2	5	10		-					
…	…	2	4	8		-					
…	…	1	4	4		-					

Figure 13: Example of a Risk Plan Matrix for an IT development project

Risk Monitoring and control

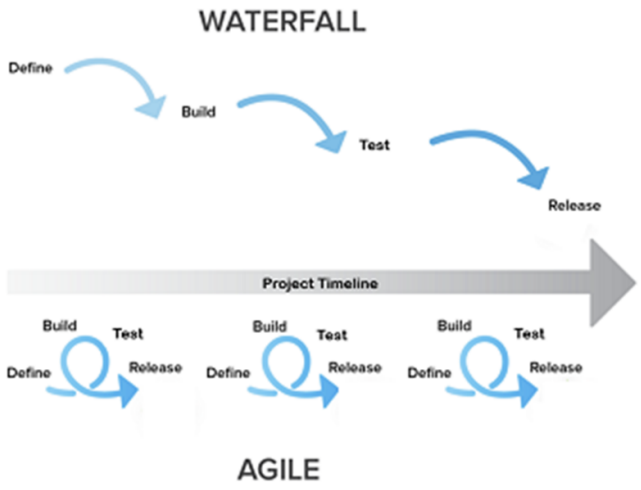
Risk monitoring is a critical aspect of IT project management as it involves continuously monitoring the project for potential risks and taking action to address them as they arise. Risk monitoring is an ongoing process that requires the project team to regularly review and update the risk management plan to ensure that it remains effective in managing project risks.

During the risk monitoring process, the project team should regularly review the project's progress and performance against the risk management plan. They should assess whether any new risks have emerged or whether the probability or impact of existing risks has changed. The team should also assess whether the risk management plan is still effective in managing project risks.

If any new risks have emerged or the probability or impact of existing risks has changed, the project team should take appropriate action to update the risk management plan. This may involve developing a new risk response plan or modifying the existing one. The project team should also ensure that the stakeholders are aware of any changes made to the risk management plan.

Agile Project management

As mentioned during the introductive part of this chapter, Agile methodologies in IT project management were developed as a response to the limitations of traditional Waterfall project management methods in those contexts in which the degree of uncertainty is relevant. The concept of Agile development emerged in the late 1990s/early 2000s, as software development teams sought new ways to improve the speed and quality of their work. At the time, traditional software development methodologies, such as the Waterfall model, were dominant. These methodologies were characterized by a linear and sequential approach to software development, in which each phase of development had to be completed before moving on to the next. However, this approach proved to be inflexible and slow, leading to delays, cost overruns, and poor-quality software. In response, a group of software developers came together to create a new approach to software development, one that prioritized collaboration, iteration, and customer satisfaction. This approach became known as Agile software development, and it has since become the standard for software development across industries and around the world. In this chapter, it will be explored the history of Agile software development, its key principles and practices, and its impact on the field of software development.



Agile Software Development is an iterative and collaborative approach to software development that emphasizes flexibility and adaptability and, in fact, one of the key differences between agile and waterfall project management is how they manage uncertainty in projects. Agile methodologies embrace uncertainty as an inherent part of the project management process. Agile teams expect that requirements will change, risks will emerge, and priorities may shift. They are adaptable and responsive to changes, and feedback from stakeholders is valued and incorporated into the project quickly. Agile teams use techniques to identify and address risks and issues early, and to continuously improve their processes and outcomes.

In contrast, traditional/waterfall methodologies assume that requirements can be fully defined upfront, and risks can be mitigated through detailed planning and documentation. However, in uncertain and complex environments, requirements may change, risks may emerge, and priorities may shift, leading to delays and increased costs. Waterfall projects may struggle to adapt to changes, and scope creep or rework may occur if requirements were not thoroughly defined at the beginning. As a result, waterfall projects may be less flexible and less responsive to changing conditions.

It is built on a set of core values that prioritize people, working software, customer collaboration, and responding to change. These values are essential to the success of Agile Software Development and have a significant impact on how software is developed, delivered, and maintained.

The four **Agile Software Development values** are the following**:**

- **Individuals and interactions over processes and tools**: this value emphasizes the importance of the human element in software development. Agile teams prioritize the needs and skills of individual team members over rigid processes and tools. This means that they value communication and collaboration, encouraging face-to-face interactions over reliance on documentation or tools. By emphasizing individuals and interactions, Agile teams can respond more effectively to the changing needs of the project and can create more innovative and creative solutions.
- **Working software over comprehensive documentation**: this value emphasizes the importance of delivering working software that meets the needs of the customer over producing extensive documentation. Agile teams prioritize rapid iteration and delivery, using frequent feedback and testing to ensure that software is functional and user-friendly. This approach enables teams to quickly identify and address issues, leading to a more reliable and efficient software development process.
- **Customer collaboration over contract negotiation**: this value emphasizes the importance of engaging with the customer throughout the development process to ensure that the software meets their needs. Agile teams prioritize collaboration, working closely with the customer to understand their requirements and provide regular updates on progress. This approach enables teams to deliver software that meets the customer's expectations, leading to greater satisfaction and improved outcomes.
- **Responding to change over following a plan**: this value emphasizes the importance of flexibility and adaptability in the software development process. Agile teams prioritize responding to changing requirements and feedback over following a fixed plan. This approach enables teams to adjust their approach as needed, leading to more effective and efficient development and delivery.

It is important to underly that the left part of the sentences is more important than the right one, but it doesn’t mean that the latter should be undervalued. In fact, we should recognize that the right side is valuable, but the left is even more valuable. These principles should be used to guide the Agile team during the project, helping them to prioritize actions and choices.

One key principle of Agile methodologies is the prioritization of customer satisfaction. Agile teams focus on delivering software that meets the needs of their customers, while also providing value to the business. To achieve this, Agile teams work closely with customers and stakeholders to understand their requirements, gather feedback, and iterate on the software throughout the development process. This enables teams to deliver software that meets the changing needs of the business and the customer, while also maintaining a high level of quality.

Another important principle of Agile methodologies is the emphasis on collaboration and communication. Agile teams work together closely, often in the same physical space, to ensure that everyone has a clear understanding of the project goals, timelines, and priorities. This close collaboration enables team members to share knowledge and expertise, resolve issues quickly, and make informed decisions that support the project's success. In addition, Agile teams communicate frequently with stakeholders and customers to gather feedback and ensure that everyone is aligned on the project's direction.

Agile methodologies also emphasize the importance of iteration and continuous improvement. Agile teams work in short cycles, called sprints, typically lasting one to four weeks. During each sprint, the team works on a set of prioritized tasks, which are then reviewed and tested at the end of the sprint. This enables the team to identify and address issues quickly, while also providing opportunities for continuous improvement.

Agile software development has gained popularity due to its focus on delivering working software quickly and adapting to changing requirements. Several agile frameworks have emerged, each with its own set of principles and practices. Some of the most common Agile frameworks are:

- Scrum: Scrum is the most widely used agile framework. It emphasizes collaboration, iterative development, and continuous feedback. Scrum includes a set of roles, ceremonies, and artifacts to guide software development. The Scrum team works in sprints, typically two to four weeks long, to deliver a potentially releasable product increment.
- Kanban: Kanban is a lean agile framework that emphasizes visualizing work and limiting work in progress. It focuses on optimizing the flow of work and minimizing waste. Kanban teams use a visual board to manage work, with columns representing different stages of the workflow.
- Scrumban: it is an agile framework that combines elements of Scrum and Kanban. It uses the Scrum ceremonies, roles, and artifacts to guide software development, but also allows for continuous flow of work like Kanban. Scrumban is ideal for teams that have already adopted Scrum but want to improve their flow of work.
- Lean Startup: Lean Startup is an agile framework that emphasizes building and testing a minimal viable product (MVP) to validate assumptions before investing significant resources. The framework emphasizes continuous learning and customer feedback to iteratively improve the product.
- Extreme Programming (XP): Extreme Programming is an agile framework that emphasizes technical practices such as test-driven development, continuous integration, and pair programming. XP teams focus on delivering high-quality software quickly through frequent releases and continuous feedback.

As can be noticed in the data of Figure 14, Agile methodologies can also be used along traditional one in Hybrid configurations.

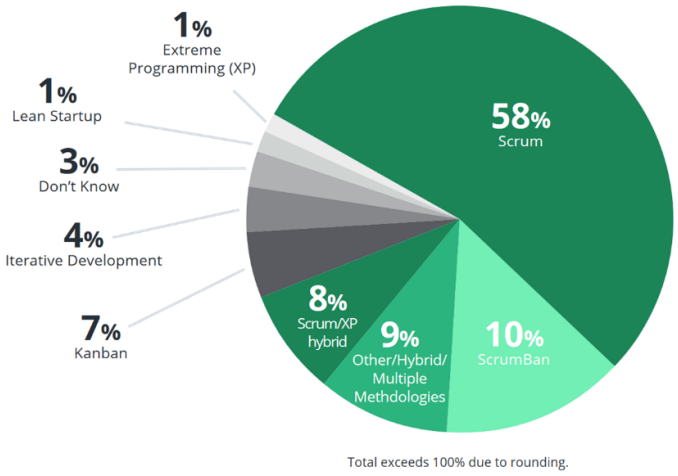


Figure 14: Diffusion of Agile methodologies as 2020, 14° annual State of Agile Report

SCRUM

Scrum is an Agile Project Management / Product Development framework that enables teams to embrace a goal-oriented mindset. Drawing inspiration from the rugby "scrum", Scrum encourages teams to approach problem-solving through experiential learning, self-organization, and reflection on past successes and failures.

Scrum relies on a set of processes and best practices that serve as tools to help teams create working software efficiently and effectively. It emphasizes that delivering results is the ultimate measure of success, and that teams must prioritize output that meets customer needs and is of high quality.

Scrum focuses on the realization of a working product, which means that the end goal is to deliver a product that meets the needs of the customer. This is achieved through a process of iterative development, where the product is developed in small increments, each of which is tested and validated before moving on to the next. This approach allows for early detection of defects and ensures that the product is continuously refined and improved over time.

Additionally, Scrum emphasizes the release of functional features at regular intervals. This approach allows for the product to be delivered to the customer in small increments, which can be tested and validated. Regular releases also provide feedback to the customer, allowing them to provide feedback on the product and identify any issues or areas for improvement. This approach ensures that the product is always evolving and improving over time and helps to ensure that it meets the needs of the customer.

As other Agile methodologies, Scrum expects changes in requirements, architecture, and design. This is because the requirements of the project can change over time, and the architecture and design of the product may need to be adapted to meet these changing requirements. Scrum is designed to be flexible and adaptable, and the iterative development approach allows for changes to be made quickly and efficiently. This means that the product can evolve over time to meet the changing needs of the customer, without requiring a complete overhaul of the development process.

Inspired by the four values of the Agile manifesto, also SCRUM defines ad first it's core values (Figure 15).

Figure 15: SCRUM Values

- Courage, which refers to the team's willingness to take risks, experiment, and be innovative. This involves stepping out of one's comfort zone and being willing to challenge the status quo, even in the face of uncertainty or adversity. By embracing a culture of courage, the team can foster creativity, problem-solving, and continuous learning.

- Focus, which refers to the team's ability to stay focused on the goals and priorities of the project. This involves being able to manage distractions, avoid scope creep, and prioritize work effectively. By embracing a culture of focus, the team can ensure that they are delivering value to the customer and achieving the project's objectives.
- Commitment, which refers to the team's willingness to take ownership of the project and to work together towards a common goal. This involves being committed to delivering high-quality work, meeting deadlines, and continuously improving the product. By embracing a culture of commitment, the team can establish trust and accountability, which are essential for success in any project.
- Respect, which refers to the team's ability to value and appreciate the contributions of each team member. This involves being respectful of each other's opinions, skills, and backgrounds, and treating everyone with dignity and professionalism. By embracing a culture of respect, the team can build a positive and supportive work environment, which is essential for achieving success in any project.
- Openness, which refers to the team's willingness to communicate openly and honestly with one another, as well as with external stakeholders. This involves being transparent about progress, challenges, and opportunities, and being receptive to feedback and constructive criticism. By embracing a culture of openness, the team can build trust and collaboration, which are critical for success in any project.

One of the key strengths of Scrum is its clear definition of roles and responsibilities, which helps teams to work collaboratively and effectively towards a shared goal. The Scrum Team is completely responsible for its output and is self-managed, which allows for greater flexibility and creativity in problem-solving. Additionally, the team is expected to deliver outputs at a predetermined pace, which helps to ensure that progress is made consistently and that the project stays on track.

SCRUM Overvew and Key Components

The SCRUM Methodology leverages on three main key components:

- Roles
- Cerimonies
- Artifacts

These three components characterize the way each iteration, that in SCRUM is called Sprint.

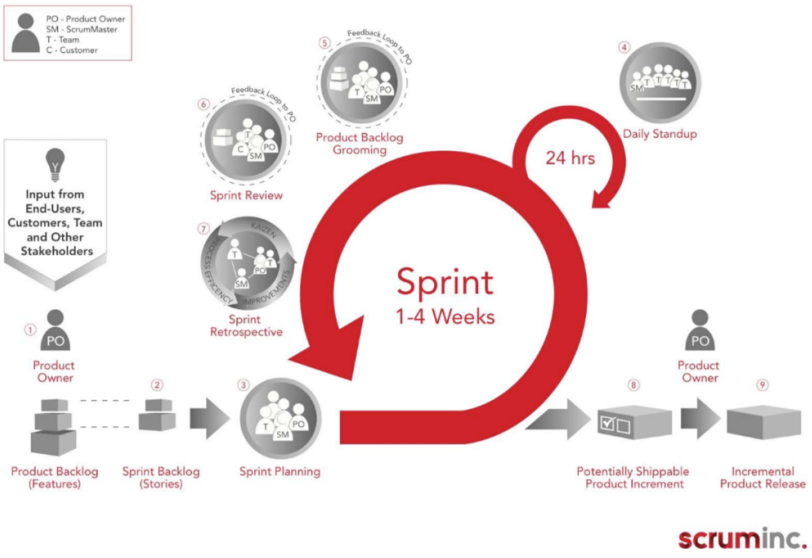


Figure 16: SCRUM Key Components

A Scrum sprint is a time-boxed period in which the Scrum team works on a set of prioritized user stories to deliver a potentially releasable product increment. The sprint typically lasts between two to four weeks and provides the team with a fixed period to work on a set of defined goals.

The sprint starts with a planning meeting where the product owner presents the sprint goal and the team reviews and selects the user stories to be worked on. The team then creates a sprint backlog, which is a list of tasks to be completed to achieve the sprint goal.

During the sprint, the team works on the tasks identified in the sprint backlog. The team collaborates closely to complete the tasks, with daily check-ins during the daily Scrum to track progress and resolve any issues or impediments that arise.

At the end of the sprint, the team presents the product increment during the sprint review meeting, where the product owner and stakeholders provide feedback on the work completed. The team also holds a sprint retrospective to reflect on the sprint process and identify areas for improvement.

Once the sprint retrospective is complete, the team starts a new sprint. The process repeats, with the team continuously working in iterative cycles to deliver value to the customer and improve their processes and practices.

In the following paragraphs each key component will be described.

SCRUM Roles

Product Owner

In the Scrum methodology, the product owner plays a crucial role in ensuring the success of the project. The product owner is responsible for defining and prioritizing the product backlog, which is a list of features and requirements that need to be implemented in the product. They are the voice of the customer and are responsible for ensuring that the product meets the needs and expectations of the stakeholders.

The product owner works closely with the Scrum team and is responsible for communicating the vision and goals of the project to the team. They are also responsible for answering any questions the team may have about the product and providing guidance on the features and requirements that are included in the backlog.

One of the key responsibilities of the product owner is to prioritize the backlog. This involves deciding which features and requirements are the most important and should be implemented first. The product owner uses a variety of factors to prioritize the backlog, including the needs of the customer, the business value of the feature, and the technical feasibility of the requirement. By prioritizing the backlog, the product owner ensures that the team is working on the most important features first, which helps to deliver value to the customer early in the project.

The product owner is also responsible for ensuring that the product backlog is up-to-date and that it accurately reflects the needs of the customer. This involves reviewing and updating the backlog on a regular basis, based on feedback from the team and the customer. By keeping the backlog up-to-date, the product owner ensures that the team is always working on the most important features and requirements.

Another important responsibility of the product owner is to accept or reject the work completed by the team. This involves reviewing the work completed during each sprint and ensuring that it meets the acceptance criteria defined for each requirement. By accepting or rejecting the work completed by the team, the product owner ensures that the product meets the requirements and expectations of the stakeholders.

In summary, the product owner is a critical role in the Scrum methodology. They are responsible for defining and prioritizing the product backlog, communicating the vision and goals of the project to the team, and

ensuring that the product meets the needs and expectations of the stakeholders. The product owner plays a key role in ensuring the success of the project and is a vital member of the Scrum team.

Scrum Master

The Scrum Master is a key role in the Scrum methodology and is responsible for ensuring the success of the Scrum team. The Scrum Master serves as a coach, facilitator, and mentor for the team, helping them to understand and adopt the principles and practices of Scrum.

One of the primary responsibilities of the Scrum Master is to facilitate the Scrum events. This includes the sprint planning, daily Scrum, sprint review, and sprint retrospective meetings. The Scrum Master ensures that these meetings are conducted effectively, and that the team is able to work together to achieve their goals.

The Scrum Master is also responsible for removing any impediments that are blocking the progress of the team. This involves identifying and resolving any issues that may be preventing the team from delivering value to the customer. The Scrum Master works closely with the product owner and the development team to ensure that any issues are addressed quickly and effectively.

Another important responsibility of the Scrum Master is to promote and facilitate continuous improvement within the team. This involves helping the team to identify areas where they can improve their processes and practices, and then working with them to implement these improvements. The Scrum Master encourages the team to experiment, learn from their experiences, and continuously improve their processes and practices.

The Scrum Master also serves as a coach and mentor for the team. They help the team to understand and adopt the values, principles, and practices of Scrum, and provide guidance on how to apply these in their work. The Scrum Master encourages the team to be self-organizing and cross-functional, and helps them to develop their skills and capabilities.

Finally, the Scrum Master is responsible for ensuring that the team is adhering to the Scrum framework and principles. They help the team to understand the roles and responsibilities of each member, and ensure that the team is following the Scrum processes and practices. The Scrum Master ensures that the team is focused on delivering value to the customer, and that they are working together effectively to achieve their goals.

Scrum/Dev Team

The Scrum Team is at the heart of the Scrum methodology and is responsible for delivering the product. It is a self-organizing and cross-functional team that works collaboratively towards a common goal. The Scrum Team is made up of individuals with different skills, backgrounds, and perspectives who work together to create a high-quality product that meets customer needs.

One of the key features of the Scrum Team is its self-organizing nature. The team is empowered to make decisions and to determine how best to accomplish its goals. This allows for greater flexibility, creativity, and agility in problem-solving, as the team can adapt and respond to changes in the project environment quickly and efficiently.

The Scrum Team is also cross-functional, meaning that it has all the skills and expertise necessary to deliver the product. This includes developers, testers, designers, analysts, and any other roles that are necessary to meet the project's objectives. By having a diverse set of skills and perspectives, the team can approach problem-solving from multiple angles, leading to better results and higher-quality outcomes.

Another important aspect of the Scrum Team is its collaborative nature. Members of the team work together closely, sharing knowledge, ideas, and feedback to achieve the project's objectives. This requires a high level of communication and transparency, as well as a willingness to learn and grow together as a team.

The Scrum Team is also responsible for delivering the product incrementally and iteratively. This involves breaking down the project into smaller, manageable pieces and delivering them in short sprints. Each sprint is a self-contained cycle that includes planning, development, testing, and review, allowing the team to receive feedback and make adjustments along the way.

SCRUM artifacts

Within Scrum, artifacts are a critical component that help teams to plan, track, and communicate progress towards their goals. Artifacts are tangible and visible outputs of the Scrum framework that provide transparency and promote collaboration among team members and stakeholders.

In this section, we will explore the concept of artifacts in the Scrum framework. We will examine the three primary artifacts in Scrum: Product Backlog, Sprint Backlog, and the Increment. We will discuss the purpose of each artifact, how they are used within Scrum, and their impact on the development process. Additionally, we will delve into the key benefits of using artifacts within Scrum, including increased transparency, improved communication, and greater accountability.

By the end of this chapter, readers will have a thorough understanding of the role of artifacts in the Scrum framework, and how they contribute to the success of Agile projects. They will have gained insights into the ways in which artifacts enable teams to work more effectively together, and to deliver high-quality products that meet the needs of their customers. Whether you are new to Scrum or an experienced practitioner, this chapter will provide valuable insights into how artifacts can be leveraged to enhance your Agile development process.

Product Backlog

The product backlog is a central artifact in the Scrum framework that defines and prioritizes the features, functions, and requirements of a product. It is essentially a living document that evolves throughout the development process and serves as the single source of truth for the development team and stakeholders.

The product backlog is owned and maintained by the product owner, who is responsible for ensuring that it accurately reflects the needs of the stakeholders and customers. The product owner is also responsible for prioritizing the items in the backlog based on their relative value and importance.

The product backlog typically consists of user stories or features that describe the functionality and requirements of the product being developed. User stories are short, simple descriptions of a feature or functionality from the perspective of the end user, and they typically follow a specific format that includes a description of the user, the user's goal, and the benefit to the user.

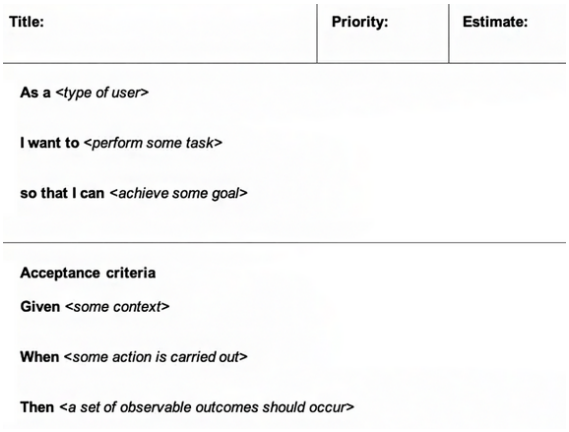


Figure 17: Example of a User Story

For example, a user story for an e-commerce website might be "As a customer, I want to be able to view my order history so that I can track my purchases and review my previous orders." This user story identifies the user (a customer), the user's goal (to view their order history), and the benefit to the user (the ability to track purchases and review previous orders).

The acceptance criteria of a user story are a set of conditions that must be met in order for the story to be considered complete and accepted by the product owner. Acceptance criteria are an important part of user story development, as they help to ensure that the development team understands the requirements of the story and can deliver a product that meets the needs of the user and the business.

Acceptance criteria are typically written in a specific format that includes a description of the expected behaviour or outcome, along with any specific requirements or constraints that must be met. For example, the acceptance criteria for the e-commerce website user story mentioned earlier might include:

- The order history page must display all previous orders placed by the customer.
- The order history page must include the date, order number, and status of each order.
- The order history page must allow the customer to sort orders by date, order number, or status.
- The order history page must be accessible from the customer account dashboard.

Acceptance criteria should be specific, measurable, and verifiable. This means that they should be clear and concise, with no room for ambiguity or interpretation. They should also be testable, so that the development team can verify that the acceptance criteria have been met before considering the user story complete.

The acceptance criteria for a user story are typically developed collaboratively by the product owner and the development team, with input from stakeholders as needed. The criteria are added to the user story and are used to guide development and testing throughout the sprint. If the acceptance criteria are not met during the sprint, the user story may be considered incomplete and may need to be carried over to a future sprint.

In addition to user stories, the product backlog may also include technical stories or infrastructure-related items, such as updates to the database or improvements to the user interface. These items are typically described in technical language and may not follow the user story format. Technical and architectural stories play an important role in the early phases of the software development because they create the “skeleton” of the future digital product.

The items in the product backlog are prioritized based on their relative value and importance to the product owner and stakeholders. The product owner is responsible for ensuring that the backlog reflects the needs and priorities of the stakeholders and that the items are ordered in a way that maximizes the value delivered by the development team.

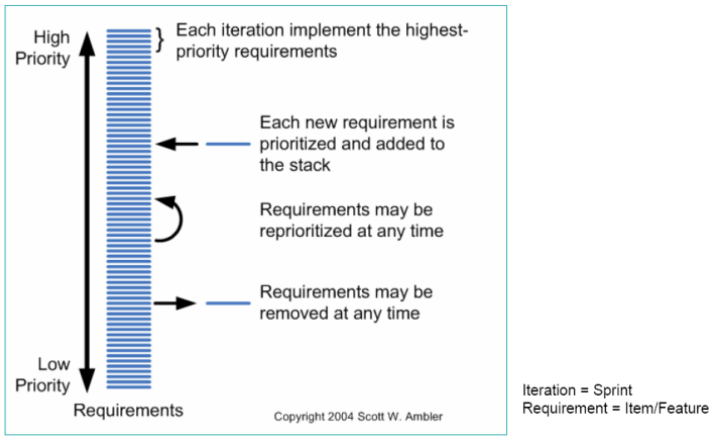


Figure 18: Product backlog management

One of the key benefits of the product backlog is that it provides a single source of truth for the entire development team. It ensures that everyone is working from the same set of requirements and priorities, which promotes alignment and reduces confusion. Additionally, because the product backlog is continuously updated and refined, it allows the development team to adapt to changing requirements and priorities, which is essential in Agile development. By providing a clear and transparent view of the development process, the product backlog enables stakeholders to provide feedback and make informed decisions about the direction of the project.

During the Sprint planning meeting, the development team reviews the items in the product backlog and selects the items that they will work on during the upcoming sprint. The selected items are then moved from the product backlog to the sprint backlog.

Sprint Backlog

The sprint backlog is a vital artifact in the Scrum framework that represents the work to be completed by the development team during a sprint. It is derived from the product backlog and serves as a dynamic plan that guides the team's activities throughout the sprint.

The sprint backlog is created during the sprint planning meeting, where the development team selects a set of user stories or product backlog items from the product backlog to be worked on during the sprint. These selected items are then broken down into smaller tasks that are more manageable for the team to complete within the sprint's timebox.

The sprint backlog typically includes the following elements:

- User Stories: The user stories or product backlog items that have been selected for the sprint are included in the sprint backlog. These stories represent the features or functionality that the team aims to deliver by the end of the sprint.
- Tasks: Each user story is broken down into smaller tasks or sub-tasks that define the specific activities required to complete the story. These tasks can include design, coding, testing, documentation, and any other necessary activities.
- Estimates: The development team provides estimates for each task, indicating the amount of effort or time required to complete it. This helps in planning and tracking progress throughout the sprint.
- Prioritization: The tasks in the sprint backlog are prioritized based on their dependencies, urgency, and their contribution to meeting the sprint goals. This ensures that the team focuses on the most important and valuable tasks first.

- Sprint Goal: The sprint backlog is aligned with a sprint goal, which is a short and concise statement that describes the objective or desired outcome of the sprint. The sprint goal helps guide the team's efforts and ensures they are working towards a common purpose.

Throughout the sprint, the development team collaborates and self-organizes to execute the tasks defined in the sprint backlog. They regularly update the status of tasks, track their progress, and make any necessary adjustments to keep the sprint on track.

The sprint backlog is a dynamic artifact that can evolve as the team learns and adapts during the sprint. New tasks may be added, existing tasks may be modified or removed, and priorities may shift based on the team's progress and feedback received. It is very important to avoid confusion between stories and tasks. While once the user stories for a specific sprint are selected it’s not possible to modify them, on the other hand is it possible (and often necessary) to add/remove tasks related to these stories during the sprint.

By maintaining a clear and well-structured sprint backlog, the development team gains transparency into their work, enabling effective planning, tracking, and collaboration. The sprint backlog serves as a powerful tool for the team to stay focused, deliver value incrementally, and achieve the sprint goal within the defined timebox.

Transparency is achieved through the visibility provided by the sprint backlog. The team members can see the tasks assigned to them, the estimated effort required for each task, and the overall progress of the sprint. This transparency allows for better coordination and understanding among team members, making it easier to identify potential bottlenecks or areas where additional support may be required.

The sprint backlog also facilitates effective planning. The tasks within the backlog are prioritized based on their value and dependencies, allowing the team to work on the most critical and impactful items first. By having a clear understanding of the work to be done, the team can allocate their time and resources appropriately, ensuring that they are focused on delivering the highest value increments of the product.

Tracking progress becomes more manageable with a well-maintained sprint backlog. As the team completes tasks and updates their status, it becomes evident how much work remains and whether they are on track to achieve the sprint goal. This visibility allows for early identification of any issues or delays, enabling the team to take proactive measures to address them and maintain the desired pace of progress.

Collaboration is enhanced through the sprint backlog as it provides a shared understanding of the work to be accomplished. The backlog serves as a common reference point for the entire team, facilitating discussions, clarifications, and decision-making. It promotes collaboration within the team and encourages collective ownership of the sprint goals, fostering a sense of unity and shared responsibility.

Furthermore, the sprint backlog acts as a source of motivation and focus for the development team. By breaking down the selected user stories into smaller, actionable tasks, the team can witness their progress and achievements as they complete each task. This incremental approach helps maintain momentum and enthusiasm, ensuring that the team remains engaged and committed to delivering value throughout the sprint.

Product Increment

The product increment is a crucial concept in the Scrum framework that represents the tangible outcome of a sprint. It is the sum of all the completed and potentially releasable product backlog items that the development team has delivered at the end of a sprint.

The product increment serves as a measure of progress and value creation. It represents the incremental development and enhancement of the product with each sprint. At the end of every sprint, the goal is to have a potentially shippable increment that can be released to customers or stakeholders if desired. It

demonstrates the continuous delivery of value to the stakeholders and reflects the collective effort and collaboration of the development team in transforming user stories or product backlog items into a working and valuable product. Each increment builds upon the previous one, gradually adding new features, improvements, and functionalities.

The product increment is expected to meet the DoD, Definition of Done (previously described also as Acceptance Criteria), which is a set of predetermined criteria that define the quality standards and completeness of a product backlog item. The DoD ensures that each increment is of high quality, free of defects, and ready for immediate use or release (so, acceptable by the PO and the stakeholders).

The product increment is reviewed during the sprint review meeting, where the Product Owner (as a representative of the stakeholders) provides feedback, offer suggestions, and assess whether the increment meets the expectations. This not only provides an opportunity for stakeholders to see tangible progress and assess the evolving product, but, moreover, the product increment facilitates the incremental delivery of value, enabling the stakeholders to realize benefits earlier. By delivering functional and potentially releasable increments at regular intervals, the development team can provide value to the customers or end-users even before the completion of the entire project.

The concept of the product increment aligns with the Agile principles of delivering working software frequently and satisfying the customer through early and continuous delivery. It emphasizes the iterative and incremental nature of Agile development, allowing for flexibility, adaptation, and feedback-driven improvements.

SCRUM Ceremonies

SCRUM ceremonies play a vital role in promoting collaboration, transparency, and effective communication among team members and stakeholders. These structured meetings provide essential opportunities for the team to plan, synchronize, inspect, and adapt throughout the project's lifecycle, as they serve as key milestones within each sprint, ensuring that the team stays focused, aligned, and responsive to changing requirements and priorities. They provide dedicated time and space for different stakeholders to come together, exchange information, and make informed decisions.

One of the primary benefits of Scrum ceremonies is the facilitation of communication and collaboration within the development team. These meetings create a platform where team members can openly discuss progress, challenges, and ideas. By fostering a collaborative environment, Scrum ceremonies enable the team to work together more effectively and leverage their collective knowledge and skills (People and interactions over processes and tools).

Furthermore, Scrum ceremonies promote transparency by making progress visible to both the team and stakeholders. They offer regular checkpoints for reviewing the work completed, identifying any deviations or impediments, and gathering feedback. This transparency ensures that all team members and stakeholders are informed about the project's status and can make informed decisions based on real-time information.

Scrum ceremonies are also essential to the iterative and incremental nature of the Scrum framework. By providing dedicated time for planning, review, reflection, and adaptation, these ceremonies enable the team to continuously improve their processes and deliver value incrementally. They create a rhythm and cadence that allows for regular feedback loops and course corrections.

Ultimately, Scrum ceremonies are essential for maximizing the benefits of the Scrum framework. They provide the structure and discipline necessary to ensure effective collaboration, transparency, and alignment within the team and with stakeholders. By actively engaging in these ceremonies, the team can enhance their communication, make timely decisions, and continuously improve their work, leading to successful project outcomes.

The main SCRUM Ceremonies are:

- Sprint Planning meeting, where the product owner and development team collaborate to define the sprint goal and select user stories or backlog items for the upcoming sprint. The meeting emphasizes the importance of clear goals, prioritization, and commitment to delivering value.
- Daily SCRUM meeting, a brief, daily meeting where team members synchronize their activities, share progress, and identify any impediments, highlighting the importance of focus, transparency, and accountability in these daily stand-ups
- Sprint Review meeting, a dedicated moment for the development team to showcase the completed product increment to stakeholders. Feedback and insights provided during this meeting contribute to continuous improvement and help align the product with customer expectations
- Sprint Retrospective, a reflection and improvement session at the end of each sprint. The chapter emphasizes the significance of an open and collaborative environment where the team can identify successes, challenges, and areas for improvement to enhance their processes.

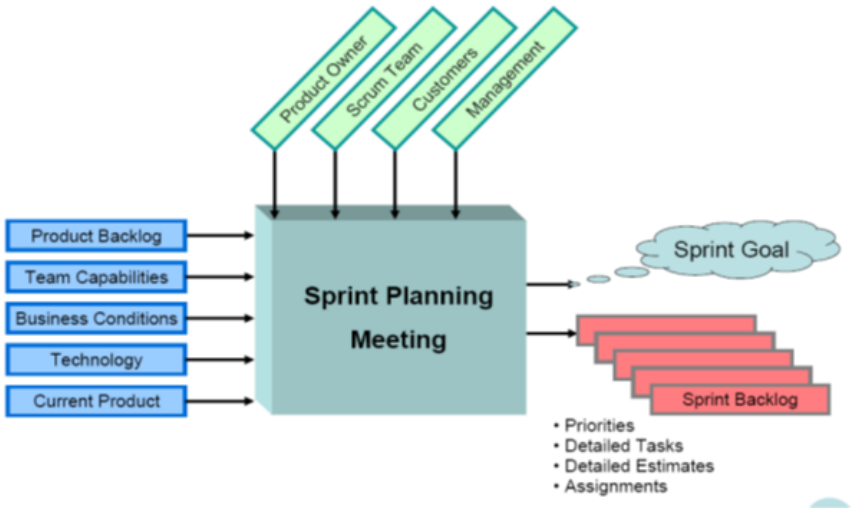
Sprint Planning

The Sprint Planning meeting is a crucial ceremony in the Scrum framework that sets the stage for a successful sprint; it is performed at the beginning of every sprint, so usually it takes place every 7 to 21 days, accordingly to the sprint duration defined for that specific project. This meeting brings together the product owner and the development team to collaboratively define the sprint goal and select the user stories or backlog items that will be worked on during the sprint. During the Sprint Planning meeting, the product owner shares the product backlog items that are of highest priority and provides insights into their expectations and requirements. The development team actively participates in the discussion, seeking clarification and expressing any concerns or challenges they foresee. This collaborative dialogue helps ensure a shared understanding of the desired outcomes and facilitates alignment between the product owner and the development team.

One of the primary objectives of the Sprint Planning meeting is to define a clear sprint goal. The sprint goal serves as a guiding principle for the development team, providing focus and direction throughout the sprint. It helps the team make informed decisions about which user stories or backlog items (i.e. technical stories) to select, ensuring that their efforts are aligned with the overall project objectives.

Another key aspect of the Sprint Planning meeting is the selection and estimation of the user stories or backlog items that will be included in the sprint. The development team examines the product backlog and, based on the sprint goal and their capacity, determines the user stories or backlog items that can be realistically completed within the sprint. The team collaboratively estimates the effort required for each selected item, using techniques such as story points (i.e. Planning Poker technique) or relative sizing.

The outcome of the Sprint Planning meeting is a well-defined sprint backlog, consisting of the selected user stories or backlog items that will be the focus of the team's efforts during the sprint. The sprint backlog becomes the basis for tracking progress and provides visibility into the work that needs to be accomplished.



Effective Sprint Planning meetings promote collaboration, shared understanding, and commitment among team members. They set the stage for a productive and focused sprint, with a clear sprint goal and a well-defined sprint backlog. This ceremony allows the development team to plan their work and make informed decisions, fostering a sense of ownership and empowerment. The Sprint Planning meeting also contributes to transparency by providing stakeholders with insights into the team's priorities and planned work for the sprint. It helps manage expectations and ensures that everyone is on the same page regarding the sprint's objectives and deliverables.

Daily Meeting

The Daily Scrum, also known as the daily stand-up, is a fundamental ceremony in the Scrum framework that enables effective communication, coordination, and synchronization within the development team. This short and focused meeting, held at the same time and place every day, serves as a valuable opportunity for team members to align their efforts and ensure progress towards the sprint goal.

The primary objective of the Daily Scrum is to promote transparency and keep the entire team informed about the progress made since the last meeting, the work planned for the day, and any potential obstacles or impediments. Each team member provides a brief update, answering three key questions: What did I accomplish yesterday? What am I planning to do today? Are there any blockers or challenges?

By sharing this information, team members gain visibility into each other's work and can identify areas of collaboration or potential dependencies. This promotes a sense of shared responsibility and helps avoid duplicating efforts or conflicts. It also provides an opportunity to identify and address any obstacles or challenges that may be hindering progress, allowing the team to find solutions and keep the momentum going.

The Daily Scrum is typically time-boxed to around 15 minutes to ensure a focused and efficient meeting. It is essential that all team members actively participate and stay engaged throughout the meeting. To maintain the brevity and effectiveness of the Daily Scrum, discussions about detailed solutions or extensive problem-solving are deferred to separate discussions outside of the meeting.

This daily synchronization fosters a sense of unity and shared purpose within the team. It encourages open communication and trust, enabling team members to collaborate effectively and leverage each other's skills and expertise. By staying informed about each other's progress and challenges, team members can provide support, offer suggestions, or seek assistance when needed.

Moreover, the Daily Scrum helps identify any potential misalignments or deviations from the sprint goal early on. If a team member's update reveals a potential divergence or impediment, it can be promptly addressed, allowing the team to adjust their plans and mitigate any risks or issues before they escalate.

The Daily Scrum also serves as a valuable accountability tool. By providing a regular forum for progress updates, team members are encouraged to stay committed to their individual tasks and responsibilities. This promotes a sense of ownership and enables the team to collectively assess their progress and take corrective actions as needed.

Sprint Review

The Sprint Review meeting is the ceremony that allows the development team to showcase their completed work to stakeholders and obtain valuable feedback. This collaborative session, held at the end of each sprint, serves as an opportunity to evaluate the product increment and ensure alignment with customer expectations.

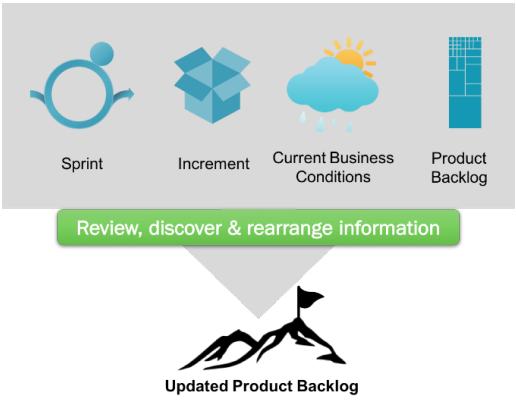


Figure 19: main aspects of the sprint review (source Scrum.org)

During the Sprint Review, the development team presents the functionality and features that have been accomplished during the sprint. This demonstration provides the product owner, and eventually other stakeholders, with a tangible representation of the progress made. It allows them to interact with the product increment and gain a better understanding of its capabilities.

The Sprint Review meeting is not just a one-way presentation; it is an interactive session that encourages active participation and feedback from stakeholders. They have the opportunity to ask questions, provide suggestions, and express their opinions on the increment's functionality and usability. This feedback is invaluable in shaping the product's direction and ensuring it aligns with customer needs.

Another important aspect of the Sprint Review meeting is the validation of the product increment against the previously defined sprint goal. The team evaluates whether they have successfully achieved the sprint goal and met the agreed-upon acceptance criteria. This evaluation provides a clear assessment of the team's progress and serves as a basis for improvement in subsequent sprints.

The Sprint Review meeting promotes transparency by providing stakeholders with visibility into the product development process. It allows them to assess the work completed, offer insights, and make informed decisions based on the current state of the product. This transparency helps build trust and fosters a collaborative relationship between the development team and stakeholders.

Additionally, the Sprint Review meeting acts as a platform for continuous improvement. The feedback and insights gathered during the session contribute to the team's learning and enable them to refine their future

work. It allows for the identification of areas of improvement, potential enhancements, and adjustments to better align the product with customer expectations.

The Sprint Review meeting not only benefits the development team and stakeholders but also serves as a celebration of the team's achievements. It acknowledges the hard work and dedication of the team members and provides a sense of accomplishment as they demonstrate the value they have delivered. This recognition boosts morale and motivates the team for future sprints.

Monitoring and controlling the project

As mentioned in the previous paragraphs, the sprint review meeting provides an opportunity to evaluate the progress of the project, in terms of potentially shippable product, showcasing the completed product increment to the stakeholders.

In addition, another primary way Scrum tracks progress is with the burndown chart. The burndown chart is an essential tool in Scrum for tracking progress that visually represents the amount of work remaining over time, allowing the team to monitor their progress and adjust their efforts accordingly. The burndown chart shows the ideal progress line, which represents the planned completion of all tasks by the end of the sprint, as well as the actual progress line, which shows the real-time progress made. By comparing the two lines, the team can identify any deviations and take corrective actions to stay on track.

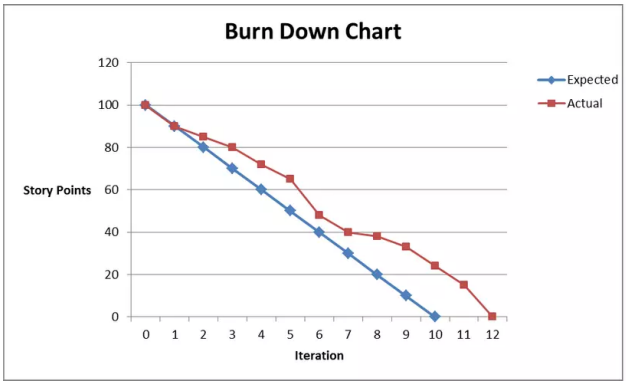


Figure 20: Example of a Project Burndown Chart

A burndown chart visually represents the remaining work in a sprint over time. The chart typically consists of two axes: the vertical axis represents the amount of work remaining, and the horizontal axis represents the sprint's duration in time (e.g., days or weeks). As the team completes tasks or user stories, the burndown chart shows a downward trend, indicating progress. It helps stakeholders and the team visualize the overall sprint progress and identify any potential delays or deviations from the planned work.

The amount of work completed by the development team in each sprint, instead, it’s called velocity. It is calculated by summing up the story points or effort estimates for the user stories or backlog items successfully delivered within a sprint. Velocity provides an objective measure of the team's capacity and productivity. By tracking the team's velocity over multiple sprints, it becomes a reliable metric for forecasting and planning future work. It helps the team establish a predictable pace and set realistic goals for future sprints.

During the Sprint Review meeting, the burndown chart and velocity are used to facilitate discussions and evaluate the team's performance. The burndown chart offers a visual representation of the sprint progress, enabling the team to assess whether they are on track to achieve their sprint goals. It helps identify any potential bottlenecks or issues that may have affected progress and allows the team to reflect on the reasons behind deviations from the expected trajectory. This information becomes a valuable input for discussions during the review meeting, enabling the team to analyze the factors that influenced their performance and make informed decisions for future sprints. Velocity, on the other hand, provides a quantitative measure of

the team's productivity and capacity. It helps stakeholders and the team evaluate the team's ability to consistently deliver work within a sprint. By comparing the team's velocity to their initial estimates or commitments, the review meeting becomes an opportunity to assess whether the team is accurately estimating their capacity and adjusting their plans accordingly. It allows for meaningful conversations about balancing workload, adjusting expectations, and ensuring the team has a realistic understanding of their capabilities.

Sprint Retrospective

The Sprint Retrospective meeting is a crucial ceremony in the Scrum framework that enables the development team to reflect on their performance, identify areas for improvement, and make necessary adjustments for future sprints. This collaborative session, held at the end of each sprint after the Sprint Review, focuses on learning from past experiences to enhance team effectiveness and optimize the development process.

During the Sprint Retrospective, team members engage in an open and honest discussion about what went well, what could be improved, and how to address any challenges or issues encountered during the sprint. The retrospective creates a safe environment where team members can freely express their opinions, share insights, and contribute to the collective learning process.

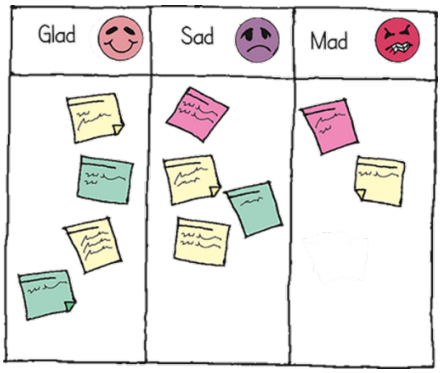


Figure 21: Example of a Glad, Sad,Mad visual board

The "Glad, Sad, Mad" methodology is a simple and effective technique used in Sprint Retrospectives to gather feedback and insights from team members. Team members share their observations and feelings regarding the sprint using three categories: "Glad" (positive aspects), "Sad" (concerns or disappointments), and "Mad" (frustrations or issues), then they start a discussion about all the aspects they identified.

Once all perspectives have been shared and discussed, the team collectively moves towards identifying improvement actions. Building upon the insights gained from the "Glad, Sad, Mad" exercise, team members are encouraged to generate ideas and suggestions for addressing their concerns, capitalizing on the positive aspects, and alleviating any frustrations. They engage in a collaborative process to prioritize the most relevant improvement actions, ensuring that decisions reflect their shared goals and objectives. This inclusive approach empowers the team to take ownership of their improvement journey and select actions that resonate with their collective vision and capabilities. By actively participating in this process, the team cultivates a sense of ownership and accountability, driving meaningful changes and continuously enhancing their performance and effectiveness.

The retrospective meeting emphasizes the importance of continuous improvement and encourages the development team to take ownership of their processes. By reflecting on their work, the team can identify patterns, evaluate the effectiveness of their practices, and determine appropriate actions to enhance productivity and collaboration in subsequent sprints.

To ensure the retrospective meeting is effective, it is important to focus on actionable outcomes. The team collaboratively identifies specific improvement actions and assigns responsibilities for their implementation. These action items are documented and reviewed in subsequent retrospectives to track progress and ensure accountability.

Chapter 14

Ethical issues

Part of this chapter is derived from [15], only for internal class use, not to be redistributed.

Ethical and Social Issues and regulations in Digital Systems

Part of this material is based on Laudon& Laudon, Management Information Systems, Chapter 4 [

1

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

LEARNING OBJECTIVES

- What ethical, social, and political issues are raised by information systems?
- What specific principles for conduct can be used to guide ethical decisions?
- Why do contemporary information systems technology and the Internet pose challenges to the protection of individual privacy and intellectual property?
- How have information systems affected everyday life?

2

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Behavioral Targeting: Your Privacy Is the Target

- **Problem:** Need to efficiently target online ads.
- **Solutions:** Behavioral targeting allows businesses and organizations to more precisely target desired demographics.
  - Google uses tracking files to monitor user activity on thousands of sites; businesses monitor activity on their own sites to better understand customers.
  - Demonstrates IT's role in organizing and distributing information.
  - Illustrates the ethical questions inherent in online information gathering.

3

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Understanding Ethical and Social Issues Related to Systems

- **Ethics**
  - Principles of right and wrong that individuals, acting as free moral agents, use to make choices to guide their behaviors

4

- **Information systems and ethics**
  - **Information systems raise new ethical questions because they create opportunities for:**
    - Intense social change, threatening existing distributions of power, money, rights, and obligations
    - New kinds of crime

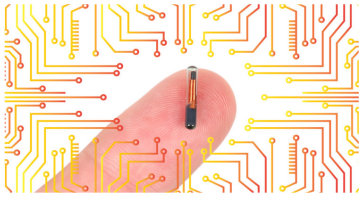
5

An example for discussion

Implantable RFIDs

https://www.youtube.com/watch?v=DSmr-yuLLr4

What if your company offers to microchip you? (2018)



6

What about other possible uses?

- https://ceo-insight.com/innovation/can-chips-improve-your-health/ (Jan. 2020)
- Example Rapid identification to access medical records in emergencies
- Voluntary vs involuntary
- Need for laws

- In 2004, the US Food and Drug Administration (FDA) approved the use of Applied Digital Systems microchip implants to store medical information. In 2018, the use of medical implanted tags offering continuous monitoring was also approved by the FDA to monitor blood glucose levels in diabetic patients.

7

Papers

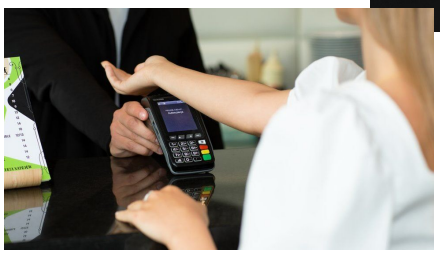
- The murky ethics of implanted chips
  IEEE Spectrum, 2007

- Chipping at work, Iowa Law Review, 2018
- https://ilr.law.uiowa.edu/print/volume-104-issue-3/chipping-in-at-work-privacy-concerns-related-to-the-use-of-body-microchip-rfid-implants-in-the-employeremployee-context/

8

Microchip implants

source: https://www.bbc.com/news/business-61008730 April 2022



9

Worried about identification?

- Think of tracking applications

10

Mobile application data

- An MIT study by de Montjoye et al. (2013) showed that 4 spatio-temporal points, approximate places and times, are enough to uniquely identify 95% of 1.5M people in a mobility database

de Montjoye, Yves-Alexandre; César A. Hidalgo; Michel Verleysen; Vincent D. Blondel (March 25, 2013). "Unique in the Crowd: The privacy bounds of human mobility". Nature srep. doi:10.1038/srep01376

11

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Understanding Ethical and Social Issues Related to Systems

- **A model for thinking about ethical, social, and political Issues**
  - **Society as a calm pond**
  - **IT as rock dropped in pond, creating ripples of new situations not covered by old rules**
  - **Social and political institutions cannot respond overnight to these ripples—it may take years to develop etiquette, expectations, laws**
    - Requires understanding of ethics to make choices in legally gray areas

12

Management Information Systems

- Ethics: individual judgement in decision making
- Society: reaching consensus on behaviors as a community
- Legal: laws are proposed, discussed, approved

13

Understanding Ethical and Social Issues Related to Systems

- Five moral dimensions of the information age:
  - Information rights and obligations
  - Property rights and obligations
  - Accountability and control
  - System quality
  - Quality of life

14

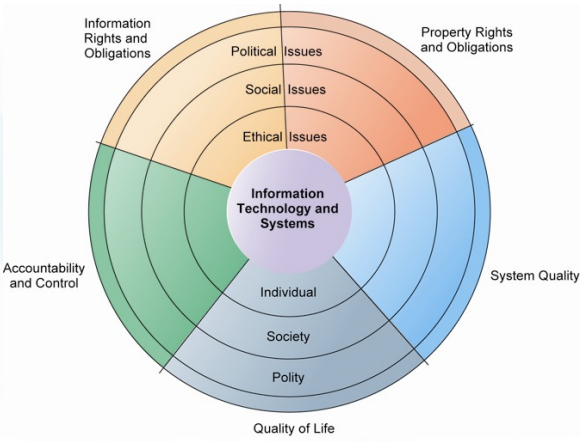
Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

THE RELATIONSHIP AMONG ETHICAL, SOCIAL, POLITICAL ISSUES IN AN INFORMATION SOCIETY

The introduction of new information technology has a ripple effect, raising new ethical, social, and political issues that must be dealt with on the individual, social, and political levels. These issues have five moral dimensions: information rights and obligations, property rights and obligations, system quality, quality of life, and accountability and control.

Figure 4-1



15

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Understanding Ethical and Social Issues Related to Systems

- **Advances in data analysis techniques**
  - Profiling
    - Combining data from multiple sources to create dossiers of detailed information on individuals
  - Nonobvious relationship awareness (NORA)
    - Combining data from multiple sources to find obscure hidden connections that might help identify criminals or terrorists
- **Mobile device growth**
  - Tracking of individual cell phones

17

ETHICAL ASPECTS

23

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Ethics in an Information Society

- **Candidate ethical principles**
  - **Golden Rule**
    - Do unto others as you would have them do unto you.
  - **Immanuel Kant's Categorical Imperative**
    - If an action is not right for everyone to take, it is not right for anyone.
  - **Descartes' Rule of Change**
    - If an action cannot be taken repeatedly, it is not right to take at all.

24

- Candidate ethical principles (cont.)
  - **Utilitarian Principle**
    - Take the action that achieves the higher or greater value.
  - **Risk Aversion Principle**
    - Take the action that produces the least harm or potential cost.
  - **Ethical "No Free Lunch" Rule**
    - Assume that virtually all tangible and intangible objects are owned by someone unless there is a specific declaration otherwise.

25

- **Five-step ethical analysis**
  1. **Identify and clearly describe the facts.**
  2. **Define the conflict or dilemma and identify the higher-order values involved.**
  3. **Identify the stakeholders.**
  4. **Identify the options that you can reasonably take.**
  5. **Identify the potential consequences of your options.**

26

- **Basic concepts for ethical analysis**
  - **Responsibility:**
    - Accepting the potential costs, duties, and obligations for decisions
  - **Accountability:**
    - Mechanisms for identifying responsible parties
  - **Liability:**
    - Permits individuals (and firms) to recover damages done to them
  - **Due process:**
    - Laws are well-known and understood, with an ability to appeal to higher authorities

27

SOCIETY

28

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

Ethics in an Information Society

- **Professional codes of conduct**
  - **Promulgated by associations of professionals**
    - Examples: AMA, ABA, AITP, ACM
  - **Promises by professions to regulate themselves in the general interest of society**
- **Real-world ethical dilemmas**
  - **One set of interests pitted against another**
    - Example: right of company to maximize productivity of workers versus workers right to use Internet for short personal tasks

4.29

29

ACM Code of ethics
Association for Computer Machinery

- https://www.acm.org/code-of-ethics

- 1. Contribute to society and human well-being.
- 2. Avoid harm to others.
- 3. Be honest and trustworthy.
- 4. Be fair and take action not to discriminate.
- 5. Honor property rights including copyrights and patent.
- 6. Give proper credit for intellectual property.
- 7. Respect the privacy of others.
- 8. Honor confidentiality.

 30

30

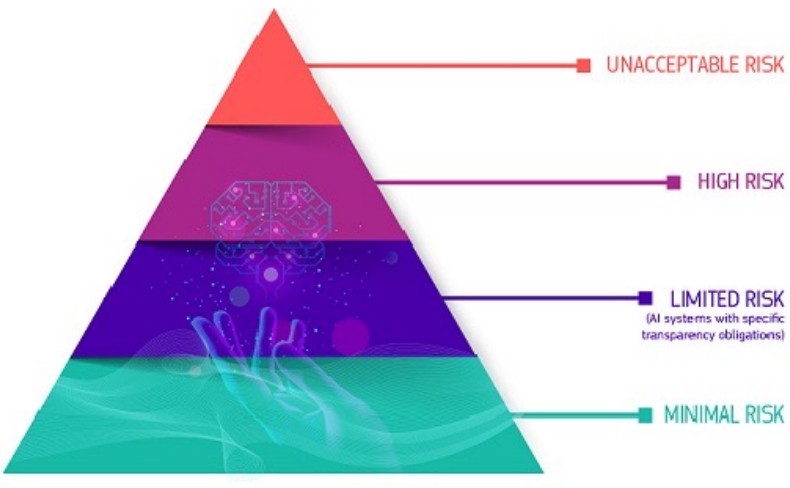
TOWRADS REGULATIONS

31

EU Commission proposes new rules and actions for excellence and trust in Artificial Intelligence –
proposed April 21, 2021

- The new **AI regulation** will make sure that Europeans can trust what AI has to offer. Proportionate and flexible rules will address the specific risks posed by AI systems and set the highest standard worldwide.
- https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai

32



 33

33

Unacceptable risk: AI systems considered a clear threat to the safety, livelihoods and rights of people **will be banned**.

High risk: AI systems will be subject to **strict obligations** before they can be put on the market.

Limited risk, i.e. AI systems with specific transparency obligations: When using AI systems such as chatbots, users should be aware that they are interacting with a machine so they can take an informed decision to continue or step back.

Minimal risk: The legal proposal allows the free use of applications such as AI-enabled video games or spam filters.

 34

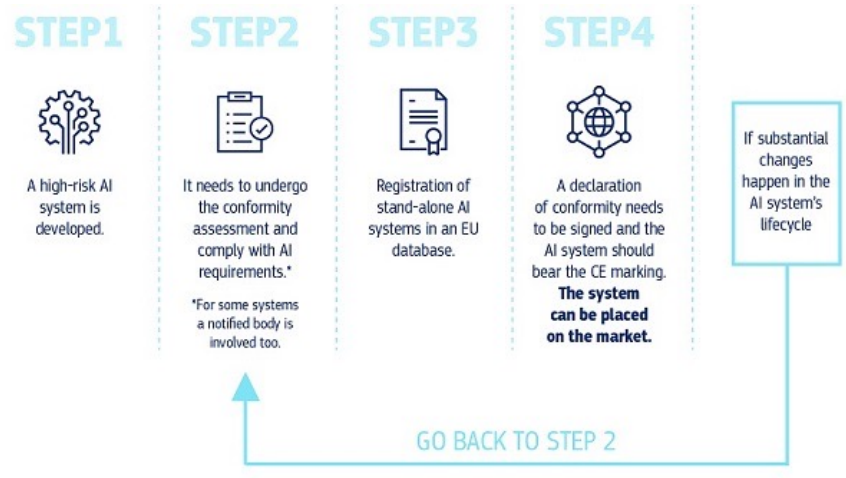
34

High risk areas

- **Critical infrastructures** (e.g. transport), that could put the life and health of citizens at risk;
- **Educational or vocational training**, that may determine the access to education and professional course of someone's life (e.g. scoring of exams);
- **Safety components of products** (e.g. AI application in robot-assisted surgery);
- **Employment, workers management and access to self-employment** (e.g. CV-sorting software for recruitment procedures);
- **Essential private and public services** (e.g. credit scoring denying citizens opportunity to obtain a loan);
- **Law enforcement** that may interfere with people's fundamental rights (e.g. evaluation of the reliability of evidence);
- **Migration, asylum and border control management** (e.g. verification of authenticity of travel documents);
- **Administration of justice** and **democratic processes** (e.g. applying the law to a concrete set of facts).

35

Actuation: Procedural aspects



 36

36

Strict obligations before they can be put on the market

- **Adequate risk assessment and mitigation systems;**
- **High quality of the datasets** feeding the system to minimise risks and discriminatory outcomes;
- **Logging of activity to ensure traceability of results**;
- **Detailed documentation** providing all information necessary on the system and its purpose for authorities to assess its compliance;
- **Clear and adequate information** to the user;
- **Appropriate human oversight** measures to minimise risk;
- High level of **robustness**, **security** and **accuracy**.

37

https://caiml.dbai.tuwien.ac.at/dighum/dighum-lectures/alexander-pretschner-software-can-do-wrong-on-ethics-in-agile-software-engineering-2023-05-16/

Software Can Do Wrong: On Ethics in Agile Software Engineering

Alexander Pretschner
Software&Systems Engineering@TUM – bidt – fortiss – CDTM

jww Jan Gogoll, Severin Kacianka, Julian Nida-Rümelin, Niina Zuber
DigHum Lectures – May 16th, 2023

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

Pretschner 2023,
Digital Humanism seminar

 38

38

Message

Context:
Digital Humanism – how „responsible“ can a machine be?
I—a software engineer—believe in personal responsibility.

Don‘t offload responsibility to others or „the system“. Yet, innovation and money matter, too.

„Ethics“ is not an AI concern only. It‘s a *software* concern. Regulation later today.

So what *could* we do?
- Education: Raise awareness. Tech students love it. Effects unclear.
- Research: Impact statements when submitting papers, IRBs or ERBs, …
- Certification: For (self-educated) software engineers? For companies?
- **Development: IRBs. Ethical Deliberation in Agile Development**
- …

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 3

Pretschner 2023,
Digital Humanism seminar

 39

39

Context

Ethics in Software Engineering

Reproach to ethicists: „Useless!“ (and to software engineers: „Not informatics anymore!“)

Indeed: >120 Codes of Conduct for AI/Software/Systems Engineering rather fruitless

Reason: Software context-specific; hence values and trade-offs context-specific
w.r.t. application domain, technology, users‘ culture, developers‘ culture, optimization goals, …

Examples: face recognition, data integration, care robots, resume analyzers, etc. –
but also software without AI: ego shooters, Corona app, BitTorrent, Telegram, Bitcoin, website preferences, ...

Genericity of CoCs hence necessary. Only way out: Deliberation schema that caters to context specificity.

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 5

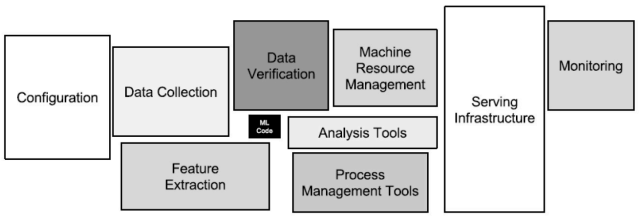
Pretschner 2023,
Digital Humanism seminar

 40

40

Software Engineering and AI

Ethical issues are not confined to AI – but this is suggested by the current debate!
A centralized Corona app? Infrastructure like Palantir Foundry? Integration of registers?



Sculley et al.: Hidden Technical Debt in ML systems, Proc. NIPS 2015: 2503–2

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 7

Pretschner 2023,
Digital Humanism seminar

 41

41

Ethical Deliberation in Agile Processes

https://www.bidt.digital/wp-content/uploads/2021/04/Digital-Transformation-and-Ethics_Zuber-et-al_EN.pdf

No simple way out. Need to address concerns in a context-specific manner: think!
Can be done in a systematic way

Development driven by EDAP:
"Ethical Deliberation in Agile [Development] Processes"

Key idea: start with and iterate on values;
continuously reflect on **mechanisms to implement them, not yes/no**

Characteristics of agility blend particularly well:
planning; incrementality; empowerment; learning

Zuber, N., *et al.* Empowered and embedded: ethics and agile processes. *Humanit Soc Sci Commun* **9**, 191 (2022)
https://www.nature.com/articles/s41599-022-01206-4

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 8

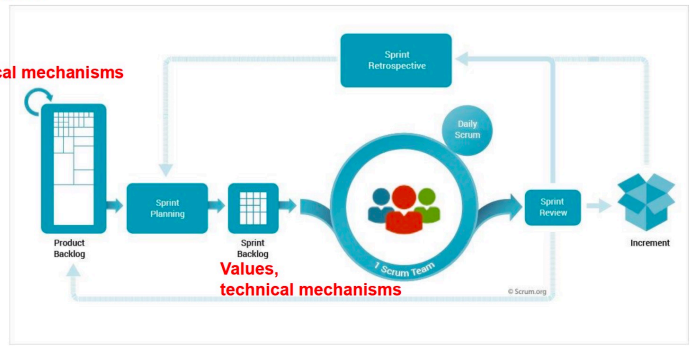
Pretschner 2023,
Digital Humanism seminar

 42

42

Scrum

Values,
socio-technical mechanisms



bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 9

Pretschner 2023,
Digital Humanism seminar

 43

43

Scrum in Context

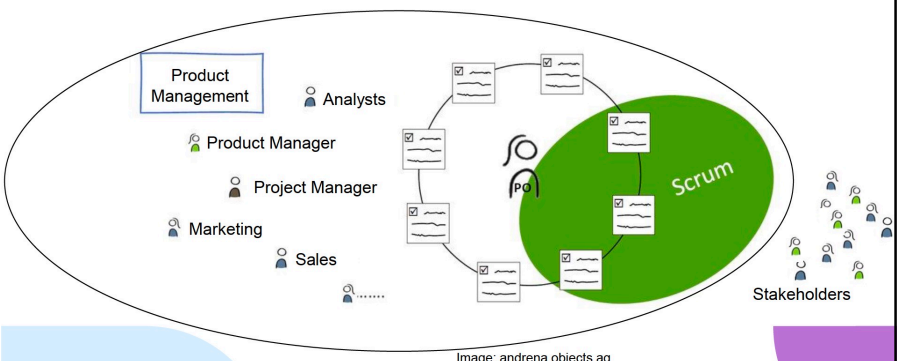


Image: andrena objects ag

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

17. Mai 2023 10

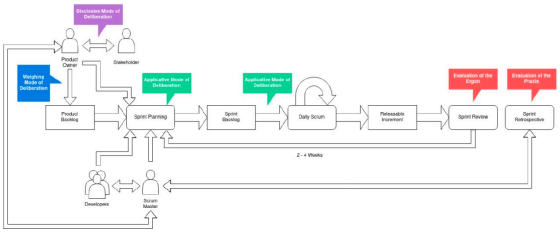
Pretschner 2023,
Digital Humanism seminar

 44

44

EDAP Ethical Deliberation in Agile Processes

Ethical Deliberation in Scrum



Techno-ethical Software Development

Episteme: analytical, hermeneutical, conceptual reasoning	Phronesis: balancing, deciding	Techne: designing and programming

Ergon and Praxis: Evaluating Products and Processes

bidt Ein Institut der Bayerischen Akademie der Wissenschaften

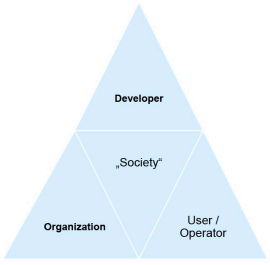
Pretschner 2023,
Digital Humanism seminar

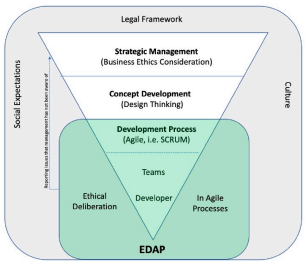
14

 45

45

Who is responsible?





Pretschner 2023,
Digital Humanism seminar

16

 46

46

LEGAL FRAMEWORKS

47

Management Information Systems

Chapter 4: Ethical and Social Issues in Information Systems

The Moral Dimensions of Information Systems

- **Information rights: privacy and freedom in the Internet age**
  - **Privacy:**
    - Claim of individuals to be left alone, free from surveillance or interference from other individuals, organizations, or state; claim to be able to control information about yourself
  - **In the United States, privacy protected by:**
    - First Amendment (freedom of speech)
    - Fourth Amendment (unreasonable search and seizure)
    - Additional federal statues (e.g., Privacy Act of 1974)

48

- **European Directive on Data Protection:**
  - **Companies must inform people information is collected and disclose how it is stored and used.**
  - **Requires informed consent of customer.**
  - **EU member nations cannot transfer personal data to countries without similar privacy protection (e.g., the United States).**
  - **U.S. businesses use *safe harbor* framework.**
    - Self-regulating policy and enforcement that meets objectives of government legislation but does not involve government regulation or enforcement.

49

Privacy

Introduction to the General Data Protection Regulation (EU)

A legal framework

https://www.garanteprivacy.it/web/garante-privacy-en/home_en

50

Regulation (EE) 2016/679:

- Entered into force in May 2016
- Applied in May 2018
- Repeals Directive 95/46/EC
- Allows Member States some flexibility
- Draft Bill for implementing certain provisions of the GDPR
- The Bill replaces Law 138(I)/2001
- Updates under discussion

52

Outlining the GDPR:

- Protection of individuals
- Free movement of data in the EU
- Balance data protection against other fundamental rights
- Balance against public and legitimate interests
- Regulatory tool

53

Where are our data?

- Collected through interactions with different organizations
  - E.g. schools, doctors, insurance companies, banks, on line shopping
- Other sources
  - Telephone
  - Social networks

54

Data Collection and Privacy

- **Monitoring software**: collecting info about employees at work
  - Screens, e-mail, number of typed characters, length of breaks, used files
- **Monitoring Web Site access**
  - Visited sites, clickstream analysis, origin of visits, used hw and sw
- **Tracking applications**
  - GPS, smartphones apps
- **Cookies**
  - Store info about users
  - Initial goal: save user's preferences, maintain sessions
  - Others: monitoring users' habits and preferences
  - Can be blocked (but service may be no longer available)

55

Privacy law

- Responsibility and accountability
- General Data Protection Regulation (GDPR) (EU)
  - it does not require any enabling legislation to be applied by national governments
  - regulates also transfer of personal data to third parties and non-EU Countries

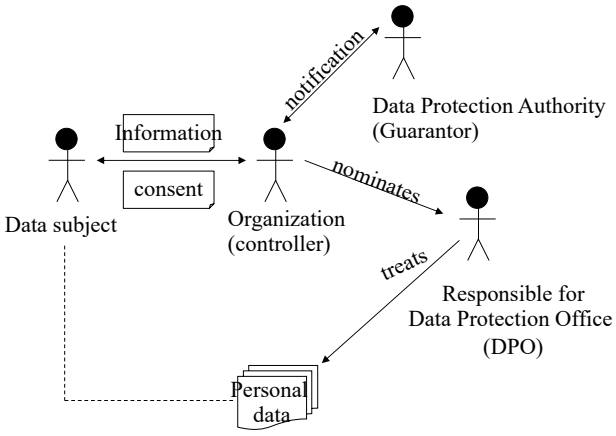
56

Some definitions

- **Personal data**: any information relating to an identified or identifiable data subject
- **Data subject**: a natural living person
- **Controller**: the owner of the data
- **Processor**: acts on behalf of the controller
- **Special categories**: health, race, religion and other special categories (**sensitive** data)
- **Processing**: collection, storage, disclosure, transfer, profiling and other processing operation
- **Profiling**: analysis/ prediction of behavior

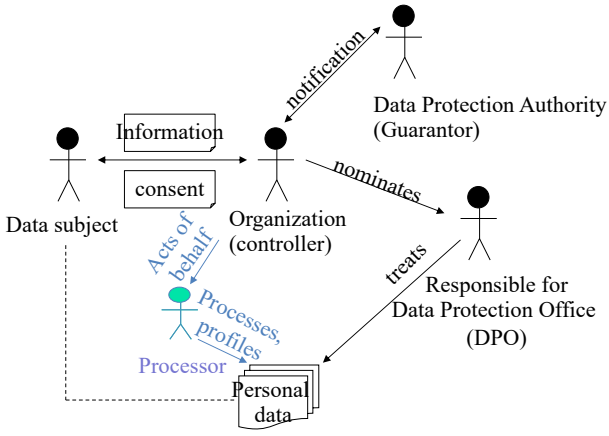
57

General principles of privacy laws



58

General principles of privacy laws



59

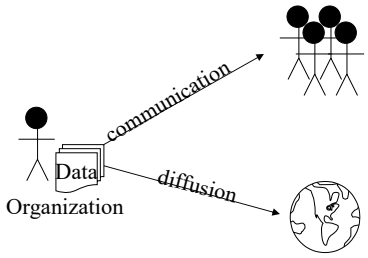
Appointment of Data Protection Officer (DPO) Article 37:

Mandatory for controller & processor when:

- Public authority or body
- Core activities consist of operations that require regular and systematic monitoring of data subjects on a large scale
- Core activities require processing of special categories of personal data or criminal convictions and offences, on a large scale

60

General principles



61

Data protection principles Article 5:

- Lawfulness, fairness and transparency
- Purpose limitation
- Data minimization
- Accuracy
- Storage limitation
- Integrity and confidentiality
- Accountability: The controller is obliged to demonstrate compliance with all of the above principles

 62

62

General principles

- **Minimality (proportionality)**
  Personal data may be processed only as long as they is adequate, relevant and not redundant wrt the purposes for which they are collected and/or further processed.
  Data must be accurate and, where necessary, kept up to date; every reasonable step must be taken to ensure that inaccurate or incomplete data, wrt the collection purposes or wrt further processing, are erased or rectified.
  Data should be kept in a form which does not allow the identification of data subjects and not for longer than necessary for collection purposes or wrt further processing.
  Member States shall put in place appropriate protection measures for personal data stored for longer periods for historical, statistical or scientific use. (art. 6).
- When **sensitive personal** data (can be: religious beliefs, political opinions, health, sexual orientation, race, membership of past organisations) are being processed, extra restrictions apply (art. 8).
- The **data subject may object** at any time to the processing of personal data for the purpose of direct marketing. (art. 14)

63

Lawfulness of processing Article 6:

- With Consent
- Without consent when:
  - Contractual obligation
  - Legal obligation
  - Vital interest
  - Public interest
  - Overriding legitimate interest

 64

64

Rights of data subjects Articles 12-22

- Transparent information provided when data collected from the person, or from a third party
- Right to access my own data (free of charge)
- Rectification of inaccurate/ incomplete data
- Erasure (right to be forgotten): when data are no longer necessary or consent is withdrawn or person objects for legitimate grounds or data have been unlawfully processed or there is legal obligation for erasure or data collected in relation to information society services

 65

65

Rights of data subjects Articles 12-22

- Restriction of processing: contested accuracy or unlawful processing or legal claims or decision is pending on exercised right to objection
- Data portability: receive data that I have given, in human or machine readable form & ask to transmit these data to other controller, when processing is based on consent or contract
- Objection (including profiling): when processing is based on public interest or legitimate interest
- Object to decisions based solely on automated processing, including profiling

 66

66

Responsibilities of controller & processor Cooperation with DPA & security Art.31,32

- The controller & the processor & where appropriate their representatives shall cooperate with the DPA, on request
- They must implement appropriate security measures, taking into account state of the art technology, costs and possible risks

 67

67

Minimal measures for digital data (1/2)

a) Authentication

b) Management of data access credentials

c) Authorization system

d) Periodic update of access rights

68

Minimal measures 2/2

e) Protection of electronic systems

f) Procedures for keeping backup copies and restoring them

g) Encryption of sensitive data

69

Responsibilities of controller & processor
Data breach notification Article 33

- The controller notifies the DPA about data breach within 72hrs unless there is no risk
- Justification required, after 72 hrs
- The processor informs the controller about data breach immediately
- The notification includes: nature of breach, number of persons affected, risks involved, measures taken or considered to be taken to mitigate risks

70

Security and cloud computing

"The government or company that manages the data, the data transfer and the data handling on the cloud must designate the cloud provider responsible for treatment"

Geographical localization constraints must be considered

71

Data management outside EU

- The privacy code prohibits in principle the "even temporary" transfer of personal data to an extra-European state if the destination or transit Country of data does not ensure an adequate level of protection
- This can happen if public cloud services are used
- The data controller must also consider the geographical location of the data

72

Bibliography

[1] Figure source to be identified.

[2] R N Anthony. *Planning and Control: a Framework for Analysis*. Cambridge MA: Harvard University Press, 1965.

[3] Gildas Avoine, Antonin Beaujeant, Julio Hernandez-Castro, Louis Demay, and Philippe Teuwen. A survey of security and privacy issues in epassport protocols. *ACM Computing Surveys (CSUR)*, 48(3):1–37, 2016.

[4] Carlo Batini and Monica Scannapieco. *Data and Information Quality - Dimensions, Principles and Techniques*. Data-Centric Systems and Applications. Springer, 2016. URL: `https://doi.org/10.1007/978-3-319-24106-7`.

[5] Francine Berman, Rob Rutenbar, Brent Hailpern, Henrik Christensen, Susan Davidson, Deborah Estrin, Michael Franklin, Margaret Martonosi, Padma Raghavan, and Victoria Stodden. Realizing the potential of data science. *Communications of the ACM*, 61(4), April 2018.

[6] Matthew Bovee, Rajendra P. Srivastava, and Brenda Mak. A conceptual framework and belief function approach to assessing overall information quality. In Elizabeth M. Pierce and Raïssa Katz-Haas, editors, *Sixth Conference on Information Quality (IQ 2001)*, pages 311–328. MIT, 2001.

[7] Arnaud Braud, Gaël Fromentoux, Benoit Radier, and Olivier Le Grand. The road to european digital sovereignty with Gaia-X and IDSA. *IEEE network*, 35(2):4–5, 2021.

[8] Wo L Chang, David Boyd, and Orit Levin. NIST Big Data Interoperability Framework: Volume 6, Reference Architecture. 2019.

[9] Edward Curry, Andreas Metzger, Arne J Berre, Andrés Monzón, and Alessandra Boggio-Marzet. A reference model for big data technologies. *The elements of big data value*, pages 127–151, 2021.

[10] C. Cappiello et al. *Fondamenti di Sistemi Informativi*. Amazon Create, 2017.

[11] Lois Evans. Protecting information assets using ISO/IEC security standards. *Information Management*, 50(6):28, 2016.

[12] Holger Funke. Automatic border control systems (egate). URL: `blog.protocolbench.org`.

[13] Sandra Geisler, Maria-Esther Vidal, Cinzia Cappiello, Bernadette Farias Lóscio, Avigdor Gal, Matthias Jarke, Maurizio Lenzerini, Paolo Missier, Boris Otto, Elda Paja, et al. Knowledge-driven data ecosystems toward data transparency. *ACM Journal of Data and Information Quality (JDIQ)*, 14(1):1–12, 2021.

[14] Theodore Johnson. Data profiling. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. `doi:10.1007/978-1-4614-8265-9\_601`.

[15] Kenneth C. Laudon and Jane P. Laudon. *Management Information Systems: Managing the Digital Firm, 15th Edition*. Pearson, 2017.

[16] Daniel L Moody and Peter Walsh. Measuring the value of information-an asset valuation approach. In *ECIS*, pages 496–512, 1999.

[17] Boris Otto. A federated infrastructure for european data spaces. *Communications of the ACM*, 65(4):44–45, 2022.

[18] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23:2000, 2000.

[19] The Hitachi Vantara Data Gaovernance Team. *Intelligent Data Governance for Dummies*. Wiley, 2018. URL: `https://www.hitachivantara.com/en-us/pdf/ebook/data-governance-for-dummies.pdf`.

[20] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *J. Manag. Inf. Syst.*, 12(4):5–33, 1996. URL: `http://www.jmis-web.org/articles/1002`.