

INFORMATION MODELLING OF ORGANIZATIONS

prof. Ing. Václav Řepa, CSc.



© Vysoká škola ekonomická v Praze, Nakladatelství Oeconomica – Praha 2021

© Václav Řepa

ISBN 978-80-245-2441-2

Table of Contents

Foreword.....	4
Chapter 1. Introduction to the field of information modelling of organizations and related aspects.....	6
1.1 Business system as an equilibrium of intentions and causality.....	6
1.2 Enterprise Architecture and MMABP Minimal Business Architecture.....	8
1.3 Process based organisation.....	13
Chapter 2. Business systems modelling.....	20
2.1 Conceptual Modelling under the object paradigm.....	23
2.2 Two times two basic dimensions of the Real World business model.....	27
2.3 Business Modelling Specification.....	31
Chapter 3. Modelling business processes.....	39
3.1 Modelling the system of business processes (Process Map).....	39
3.2 Modelling business process details.....	42
3.3 Process memory and MMABP process abstraction levels.....	62
Chapter 4. Modelling business objects.....	70
4.1 Modelling the system of business objects.....	70
4.2 Modelling business object details.....	74
Chapter 5 Consistency of business system models.....	78
5.1 Coherency of models.....	78
5.2 Structural coherency.....	80
5.3 Mutual consistency of object life cycles.....	86
5.4 Linking processes with objects.....	89
Chapter 6 Role of the information system in an organization.....	94
6.1 information system of a process driven organization.....	94
6.2 information system functionality and Data Flow Diagram.....	96
Chapter 7 Digital transformation and process-driven management.....	105
7.1 Process-driven evolution of the organization.....	107
Bibliography.....	110
Subject index.....	113
Table of Figures.....	115

Foreword

This textbook is about the information modelling of organizations. As usually in the field of informatics, the term Information Modelling has generally several different meanings. This book uses the concept in the widest sense possible. It does not express just the Information System, nor the technological view of the organization. In fact, it represents the general meaning of the term “organizational modelling” with the light flavour of informatics practices. Informatics practices influence this conception of organizational modelling in two main ways:

- Although the topic of the cognition of organizational structure and behaviour regularities traditionally belongs under management theory, informatics brings to this area the necessary precision in terms of a formal specification and systemic style of thinking. Informatics is a source of sophisticated techniques and tools, like conceptual data modelling and others, aimed at discovering the general regularities of the organizational structure and behaviour (often called “business rules”). Also, the ability to abstract all the non-content (non-informational) aspects of the organizational system is a strong tool for the essential precision of the cognition.
- Information technology is a key enabler of organizational changes, as it is established in the literature mentioning the theory of re-engineering (see Hammer M., Champy J. : “Reengineering the Corporation: A Manifesto for Business Evolution” [7], for instance). Consequently, a well designed Information System must be a clear picture of all the significant aspects of the organization (both the structural and behavioural ones). Information modelling aims to create the view of the organization that is independent of any non-content aspect (including the information technology, at first). At the same time, it is fully consistent with the process of development of the Information System (i.e., the infrastructure in general), which should naturally follow. In general, information modelling should be the first step in the process of building the infrastructures of the organization.

The book consists of seven chapters that together form one consistent text.

In the **first chapter**, we introduce the reader to the topic of information modelling of organizations. We describe the underlying principles and the methodology MMABP. We explain the principle of two-dimensional view of the business system, based on the idea of equilibrium of intentions and causality in the business system, and put this principle in the context of related aspects like Enterprise Architecture and Minimal Business Architecture. A special attention we pay to the process-driven management that generally underlies the approach of the methodology MMABP.

Second chapter explains the MMABP methodology in detail. First, we explain the general principles, which the methodology lies on, Then, we define four basic dimensions of the business system model that determine the basic types of information models. Finally, we describe the formal Business Modelling Specification as the common basis for all views of the organization occurring in the following chapters.

In the **third chapter**, we focus on the behavioural aspects of the organization. It describes the MMABP approach to the business processes analysis and design, and explains related important concepts like the process state, levels of abstraction of the process models, collaboration of processes, service-based approach to the design of the process system, etc.

In the **fourth chapter**, we focus on the causal aspects of the organization. We introduce the reader to the field of object-oriented conceptual modelling and its extension with the causality modelling by object life cycles.

Fifth chapter deals with the problem of the coherency of different models. It explains the coherency as a root of the model's quality, and introduces the concepts of completeness and correctness. Specific attention is paid to the question of "structural coherency" as the very deep level of the methodology support of the process of cognition.

Sixth chapter describes the role of the information system in the process-driven organization. In this context, we also extend the portfolio of information models with the model of the information system functionality, and explain the related basic tool: Data Flow Diagram.

In the **seventh chapter**, a short conclusion of the textbook contents is made in the context of *Digital Transformation*, followed by a general evolutionary-oriented discussion of the main possible types of approaches to organization management.

As it follows from the introductory paragraphs, besides the topic of **information modelling of organization**, this textbook also covers the field of **business process management**, *business process re-engineering*, *process-driven management* and other alternative names of this phenomenon that point to the most significant contemporary shift in the field of management theory. This topic nevertheless covers several other areas which are not usually regarded as a part of business process management or management in general, as this paradigmatic shift primarily reflects the consequences of the technology development in the fields of its use. To understand the meaning of this phenomenon one must thereby not only limit thinking to the field of technology, but to also take the field of its effects, i.e. the field of organisational management into account. On the other hand, it is impossible to understand these effects without understanding the technology which enables them. In short, **process-driven management is an interdisciplinary approach** which cannot be any different as the combination of various disciplines is the only way to understanding the essence of this phenomenon as well as to the ability to use its effects.

Chapter 1. Introduction to the field of information modelling of organizations and related aspects

The approach to the modelling of organizations presented in this textbook is based on the Methodology for Modelling and Analysis of Business Processes (MMABP). MMABP is a general methodology for modelling business systems using informatics methods and approaches. This way created model of an organization we call an *information model of an organization*. It serves as a basis for the creation of the conception of the organisation and behaviour of an organization in the form of the ontological and process-oriented models, as well as for the consequential development of its information system.

In general, the subject of interest of the information modelling of organizations is a *business system* itself. Information model of organization covers both essential dimensions of the business system: its ontological structure (what the business system consists of) as well its processes (how the actors in the business system behave). At the same time, the information model of an organization describes the business system in such a precise way, which allows its use as a first step in the process of the development of its *information system*, and consequently and in general, a full application of the possibilities of *IT in the business*. Thus, information modelling of organizations lies exactly on the border between IS/ITC and management.

1.1 Business system as an equilibrium of intentions and causality

As mentioned above, MMABP methodology is focused on the description of a **business system**. We understand the concept of the business system as **any system created and constantly developed by people, the aim of which is achieving so-called business goals**. In this conception, the business system consists of mutually collaborating business processes that are altogether focused on achieving particular business goals. Achieving of business goals is generally determined by the general rules of the environment in which the processes operate. We call the collection of those rules business rules. MMABP also pays special attention to the IS as an integral part of the business system even if it itself is a model of the business system.

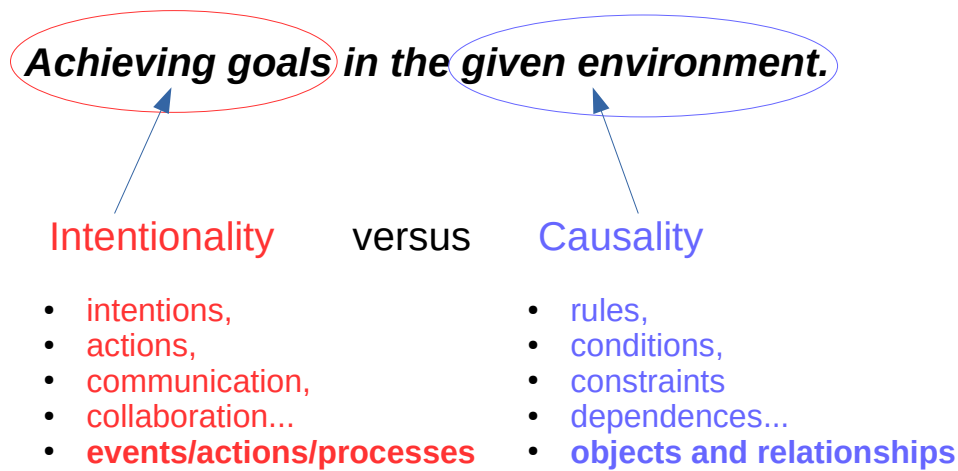


Figure 1: Business system as an equilibrium of intentions and causality

Source: Author

Figure 1 shows our idea of the essence of business: “achieving the goals in the given environment”. In this definition, two basic phenomena that form the MMABP framework for the business system modelling can be found: intentions and causality.

intentionality

The main purpose of any “business” always represents particular **intentions**, goals. The goals are achieved by actions, organized in processes that in the highly developed society, usually require collaboration since the effectiveness of the society is based mainly on its networking abilities. Consequently, collaboration requires communication. The quality and effectiveness of collaboration directly follows from the quality of communication, which opens the space for the information technology.

Regarding the intentionality of the business system, we work with **business processes** that consists of **actions** and are driven with **events**.

Causality

Besides the processes, the business system consists of people, things, their relationships, values, meanings, and other material or abstract elements, called (business) **objects**. All objects of the business system hold together by the rules, conditions, constraints, dependencies, and other kinds of their mutual **relationships**. These relationships cause the heap of business objects to be a system, to have some **logic**. Taking into account the business processes as a second essential phenomenon, the very important part of the business system logic is its **causality**, a general determination of the consequences of events, actions, and changes in the business objects.

Both phenomena are closely related since they together form the business system, and thus, they have to be harmonized. Therefore, a well tuned business system can be characterised as an **equilibrium of intentions and causality**. In the following chapters, we explain how MMABP supports this equilibrium.

1.2 Enterprise Architecture and MMABP Minimal Business Architecture

There are significant changes being made in the businesses which undergo the process of digital transformation. The reason is that the substance of digitalization is not the inherent automation, but an application of new technologies connected with a radical redesign of current business processes so that the services and products, the businesses provide to their customers, deliver much higher comfort and value. The digital transformation is a very expensive process and therefore changes and also the outcomes have to be significant otherwise it does not pay off. Simple automation of current practices or cost-saving project is not just enough [2].

The substance of the digital transformation is then in creating something that does not exist yet. The general approach to such a project is to start with a model, which allows one to test one's ideas and form something that at least works on a paper. Only when one elaborates one's ideas in a relevant formal model(s), it makes sense to start the digital transformation process, which not only includes implementation of a particular application but and mainly, the change of the business itself: business processes, products, services as well as required competences [14]. For this purpose, there have been developed modelling standards that allow one to capture the business system itself and also the information system (IS) which would support it. But the individual modelling standards are not usually enough. The model has to represent architecture built on several views (diagrams) at the modelled business system, each putting emphasis on different aspects of reality. This usually requires several standards to be used.

When creating a “vision of future reality” on paper, the mutual consistency of different diagrams, which picture the same future reality from different perspectives, is the main tool how to get to an agreement that the suggested digital transformation is feasible and makes sense at least on a paper. At the stage of analysis, this is probably the only way how to get to this conclusion.

The required speed, with which the digital transformation has to be implemented and deployed, is in the digital age relatively high [18] and there is usually not enough time to develop business architecture in such a detail that there would be some simulation possible. It would cost too much time, money and effort. Nevertheless, the completeness and correctness of the business architecture is an important issue, as any significant change in the process of implementation or even in the deployment, would endanger project delivery. The analytical stage is the place at which one should deal with it primarily. We do not suggest neglecting the detail. We just suggest that the model of the business should focus on a consistent architecture at a conceptual level and a particular detail can be elaborated in the process of implementation through, for instance, prototyping.

Unfortunately although the commonly used standards for business architecture such as TOGAF/ArchiMate, BPMN or UML provide analysts and architects with wide variety of diagrams that cover all kinds of perspectives, they are not specifying precisely what is the minimum of diagram types one has to use for a proper business architecture, which one should start with, how exactly the diagrams from different but related standards relate to each other (for instance ArchiMate and BPMN) and how their consistency can be verified.

The main goal of this chapter is to outline the minimum of mutually complementary perspectives which should be incorporated in a business architecture for digital transformation, what diagrams should be used for that and what are the consistency rules among them. Further on we illustrate how to derive requirements on IS functionality from such business architecture so that the IS functionality is fully aligned with the business architecture making so the digital transformation possible.

The content of the Minimal Business Architecture for digital transformation, which we here propose, stays at the conceptual level as we want to provide the readers with a general approach to the business system modelling, which is alterable for the individual needs of individual businesses. We do not get into an implementation detail. First, it is specific for each enterprise and its current business, application and technological architecture and second, there have been a number of methodologies already introduced for this, either traditional [3], [46], [12] or agile [1].

For the conceptual level, there are many modelling standards and frameworks available yet none of them covers the required detail of business architecture for the digital transformation completely.

We base our approach on the most popular standard for enterprise architecture i.e., TOGAF [44], which we link together with the UML [23] and BPMN [24]. We take into consideration also the ArchiMate [43] as its meta-model extends TOGAF concepts and makes them more tangible. Both TOGAF and ArchiMate standards consider the UML and BPMN as a valid extension for further architecture detail elaboration, but they do not elaborate their relation in detail. Neither does BPMN or UML. We fill these blank spaces based on the Methodology for Modelling and Analysis of Business Processes (MMABP). It not only maps the relevant relations among these standards together but also suggests a minimum of specific diagrams from these standards which should be used and put together so that one gets the complete view at the modelled business system and is able to evaluate business architecture's completeness, correctness, and mutual consistency.

A practical point of view at the business system modelling is also considered. The creation of the presented minimal business architecture would not be possible without equally capable computer-aided software engineering (CASE) tool. It has to be able to not only support the individual modelling standards used but it also has to be able to capture the relations between entities of different diagrams of different standards. In our case, we use the open-source tool Modelio and its TOGAF module, which is based on UML and covers fully all the diagrams specified in TOGAF. The interconnection between TOGAF, UML, and BPMN is in this module also included. Thanks to it, all the diagrams and models, presented in this chapter, one can create, manage and have interlinked and consistent at one place in one tool.

Core principles of Minimal Business Architecture

Unlike the modelling standards, which specify a great number of types of diagrams each applicable for a specific case, the MMABP proposes a **minimal number of types of diagrams** that have to be elaborated in order to have a compact and consistent business architecture which is ready for digital transformation.

There are two basic kinds of models of the business (Real World) system:

- **Real World ontology** represented by models expressing the important features of the business environment—modality and causality. These models are in informatics also called structural models as they are focused on the structure of the business system—which objects it consists of and how they can interact. For the ontology models MMABP uses standard modelling language UML [23], which contains diagrams for sufficient modelling of the Real World modality as well as its related causality:

- **Class Diagram** is used for the static model of the basic modality of the Real World in terms of the conceptual model. The conceptual model represents the so-called system model as it describes the whole system of mutually related business objects. This kind of description principally does not allow capturing the temporal structural aspects of the Real World (i.e., its causality) as it is focused on the common aspects of the whole system while temporal aspects are principally located just to particular elements of the system.
- **State Chart** is used for the model of the causality connected with the Real World object in terms of the so-called object life cycle. This kind of description completes the system model with temporal aspects (i.e., causality). Each model is focused on a single object and describes the causality of the Real World relevant for the given object in the form of its life cycle.

UML defines some basic relationships between both diagrams which are also a part of the MMABP rules for modelling the Real World ontology.

- **Real World behaviour** represented by models expressing the relevant behaviour of business actors as a system of business processes. Unlike in the case of the ontology, there is no integrated language in informatics for the complete description of both the system of processes and the temporal aspects of a single process. MMABP uses the standard business process modelling language **BPMN [24]** for the model of the process flow (i.e., temporal aspects) complemented with the **Process Map** for the description of the process system. Even though the Process Map is a necessary and often used type of diagram, it is not present in the BPMN standard and has to be complemented from another resource. In this chapter, we use the TOGAF Event diagram [44] for this purpose. Looking from the TOGAF perspective, the MMABP extends the TOGAF/ArchiMate business architecture by missing detail necessary for digital transformation represented by BPMN.

The above described four basic diagrams for modelling the business system the MMABP completes with the traditional diagram for the description of the needed functionality of the information system: **Data Flow Diagram (DFD)** [46]. This diagram should not be regarded as a clear model of the Real World since the functionality of the IS itself is actually a model of the Real World. Nevertheless, DFD models just the conceptual contents of the IS (its functionality) which also plays an important role in the business system. Therefore the DFD has to be also regarded as an integral part of the set of Business System model and its conceptual relationships to other models have to be defined in the methodology (see Figure 2). Moreover, taking the needed functionality of the information system as an integral part of the business system's contents supports the ideas of so-called "Digital Transformation" and "Business-IT Alignment" that characterise the current approach to the role of IT in the business.

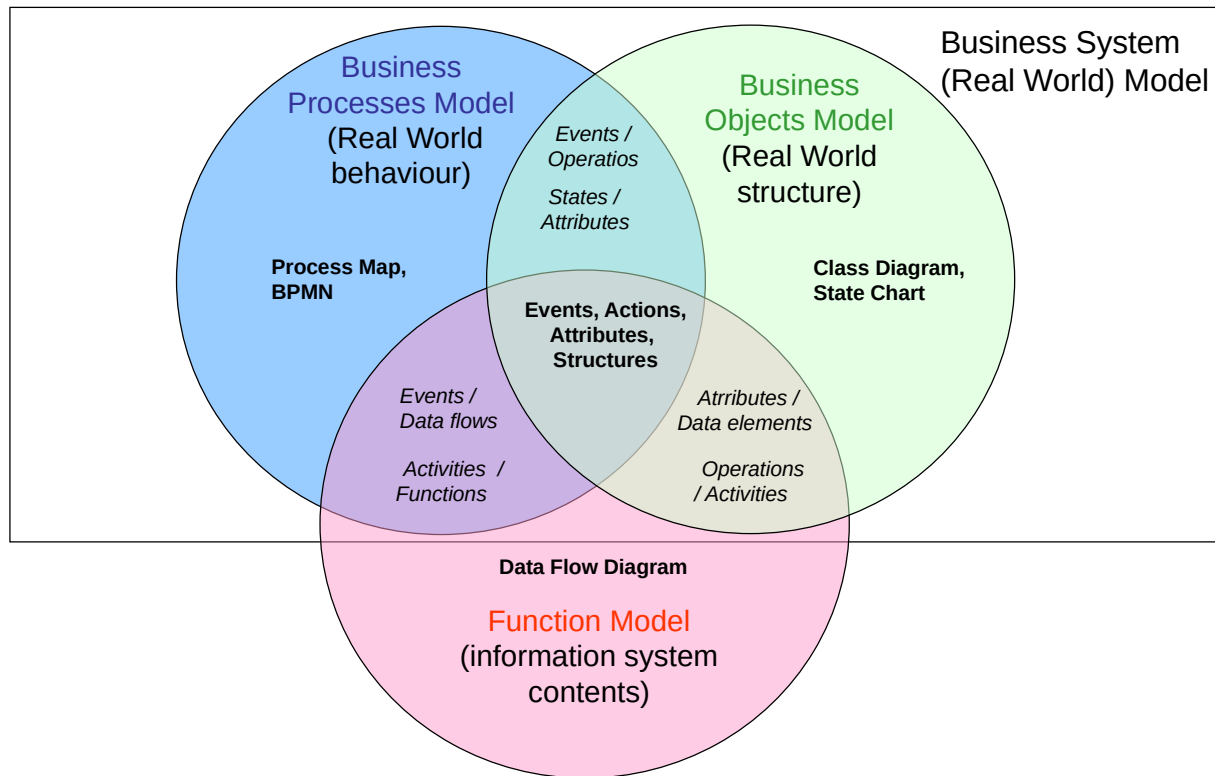


Figure 2: MMABP models

Source: Author

In both kinds of Real World models, MMABP distinguishes between two basic types of models:

- **Global (system) models** are focused on the global characteristics of the system, therefore we also call them system models. To keep the orientation on the whole system such a model has to abstract from the temporal Real World aspects. These models thus represent the naturally static view. MMABP uses the UML Class Diagram for the global ontological model and the Process Map for the global behavioural model (see above).
- **Detailed models** are focused on details of the Real World. These models are focused on temporal aspects of the Real World from both perspectives: ontological and behavioural. In order to be able to express the temporal aspects, each detailed model has to be always focused just on that part of the whole modelled system which is temporally unambiguous, i.e., it can be described as a single algorithm. Therefore, we call these models detailed. MMABP uses UML State Chart for the detailed ontological model and BPMN for the detailed behavioural model (see above).

More detailed explanation of the above-described kinds of models can be found in following chapters and a detailed description of how to proceed when modelling them in [42].

1.3 Process based organisation

During the 20 years of existence of this approach, thinking in terms of **business processes** became a regular part of organisation management practice. Nevertheless, the **Business Process re-engineering** and **Process Based Management** mean much more than it is regarded in ordinary managerial praxis. First of all it is a real paradigmatic change in the theory of management. The complexity of this shift of the paradigm makes it difficult to put into practice; moreover it is not even easy to understand the fundamental idea of this approach. Due to the facts mentioned above the full implementation of process-driven management ideas are very rare. Most stories about using process based thinking accent only marginal aspects of this approach like partial improvement of evidence, reducing time, cost, automating agendas, etc. without the real **fundamental change of business performance**, which is the real substance of the idea. On the other hand there is no business area where the implementation of Process Based Management cannot bring dramatic improvement.

Michael Hammer and James Champy [7] explain the necessity of “Business Process re-engineering” in the historical context. It shows the traditional sequence of historical milestones in the evolution of the socio-economical system from the division of labour through hand-organised production and division of management to the “Growing economy” of the 1940s to 1980s, and characterises the current situation as the “end of economic growth”. The end of economic growth has been caused by market saturation, which changes the traditional roles of the customer, co-operant, and competitor. In such situation the typical problems like hypertrophic middle management, separation of management from the customer, difficulties with clear definition of goals, heavy-handed management, problems with co-ordination of global and local goals and others are no longer acceptable for future organisation development and have to be solved unconditionally and immediately. In an unsaturated market these typical problems of a growing organisation were not critical, as it was possible to overcome them through market extension. However, thanks to the hard limits in the saturated market together with the consequential increasing pressure of competition, these problems became critical. Authors characterise the turbulent situation as “3C” (Customers, Competition, and Change) and they argue for the necessity of change in all relevant meanings; they speak about the permanent change on markets, in competition, in the nature of business, and even of the change itself – change is no longer a one-off, individual affair but the permanent attribute of the organisation.

Hammer and Champy in [7] indicate two main characteristics which should be regarded as an essence of the idea of process-oriented management.

- The main critical reason for this approach is the need for making the organisation flexible enough to be able to change its internal behaviour according to the changes in the environment. These changes not only include customer preference and requirement changes but also changes to the possibilities to satisfy them, which are typically caused by technology development.
- The main critical consequence of the above mentioned reason is the change in the concept of business organisation from strictly hierarchical to collaborative.

Once this reason is fulfilled and the organisation shifts from a formerly hierarchical to collaborative style of behaviour, the organisation can be regarded as managed in the process-oriented manner. Nevertheless, such a change requires many partial changes in all areas of the life of the organisation, where each of them can be regarded as critical. Moreover, the mutual relationships among these areas generate other consequential problems to solve. In the following text we outline and briefly discuss this complexity from the three essential perspectives.

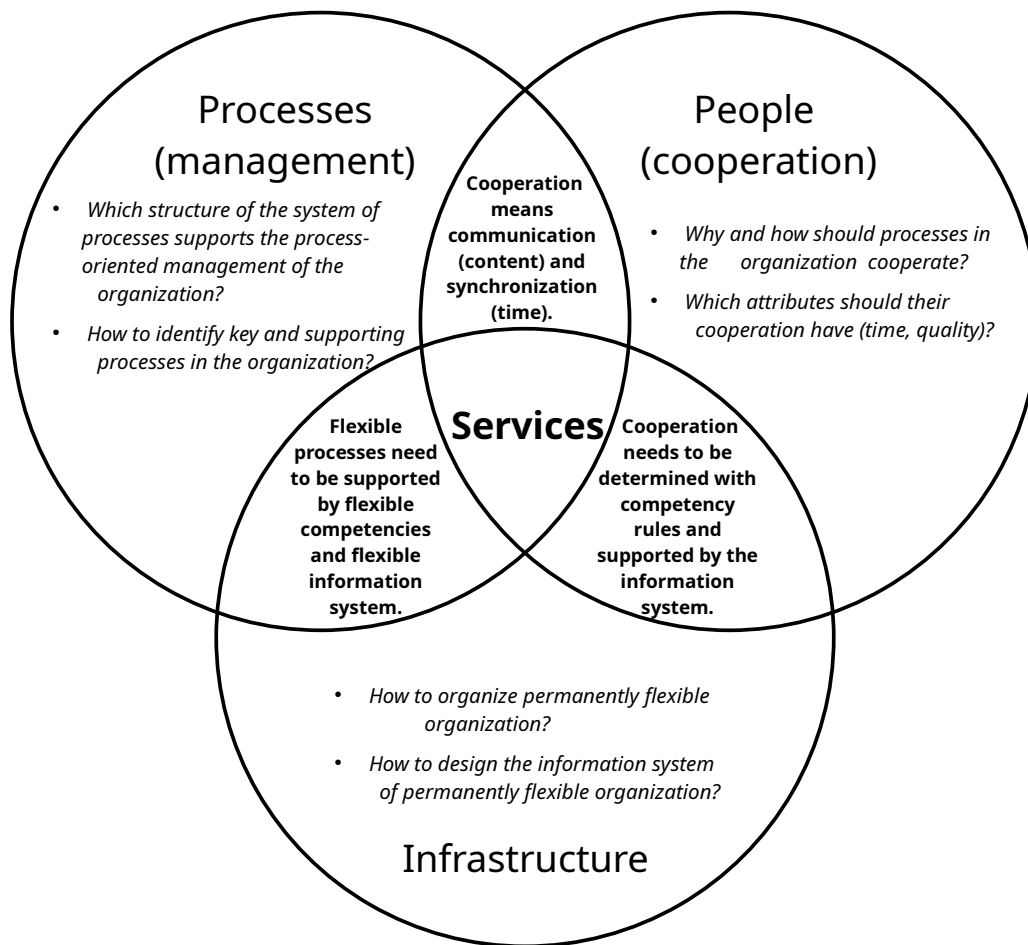


Figure 3: Content, people, and technology as three essential areas of organisation management

Source: Author

Figure 3 shows how the three essential problem areas are connected within a process based organisation. All three exemplary viewpoints are figured together addressing all substantial parts of the organisation's life: content, technology, and people. Each particular point of view is characterised by typical questions that should be answered by the methodology in that field.

Process management

Process-oriented management represents the basic idea of process based organisation, expressed excellently by Hammer and Champy in [7] and originally called “Business Process re-engineering”. This idea argues the fact that the organisation has to build its behaviour on an objectively valid structure of its business processes to be able to fully exploit the possibilities offered by the technology progress. This condition is typically not fulfilled in traditionally managed organisations where a hierarchical organisation structure prevents seeing, as well as managing, the crucial process chains which should be the central subject of change due to technology progress¹. For achieving the required ability to fully exploit the technology progress the traditional hierarchical way of management should thus be rejected and substituted with a management style based on the objectively valid model of the business processes of the organisation. Realisation of such idea nevertheless raises the consequential questions:

- Which structure of the system of processes supports the process oriented management of the organisation?
- How to identify key and supporting processes in the organisation?

To make the organisation flexible enough towards the possibilities of progress in technology, one should firstly find the “right” structural system of the organisation's business processes. It primarily means to identify the key processes profiling the organization, and accordingly, then order all necessary supporting activities into so-called supporting processes. To make the organisation flexible enough towards the possibilities of progress in technology, one should firstly find the “right” structural system of the organisation's business processes. This initially means to identify the key processes profiling the organisation, and accordingly then order all necessary supporting activities to so-called supporting processes. The key business process is such a natural process chain that covers all aspects from the initial need of a customer until the fulfilling of this requirement with the appropriate product or service.

¹ As a general example of the natural inability of the organisation to fully exploit new technological possibilities we can use database technology combined with the internet, which enables sharing of data through the organisation among its different organisational units. This possibility naturally leads to changes in the organisational structure as the existence of many organisational units is usually just a consequence of the fact that without online information sharing there is no other way of coordination of different parts of the production process (especially in territorially widespread organisations). If such organisational change is a complicated problem (as is typical in large organisations) the main power of this technology remains unexploited and the organisation is able to only use marginal, from the business point of view unimportant, aspects of the technology.

Nevertheless, the definition above does not mean that the key process has to include all of the activities necessary for the product/service delivery. It just has to cover all the process, i.e. to manage it using the services of supporting processes for ensuring the necessary productive activities/processes en route to the final delivery. In such a way the key process represents the management side of every business case, while the supporting processes represent the production side. In the process of creating the basic structure of internal business processes in the organisation via deciding on the border between the key and supporting processes the concept of Service plays the role of a universal separator. It establishes the meaning of the border between the management and the production.

The above expressed service-driven technique for creating the basic process structure of an organisation is a root idea of the MMABP methodology, which is one of the main methodical sources of the CEABPM² standards. This technique is described in more detail in [35].

Cooperation of people

Cooperation is a crucial problem in the process of building the system of processes. Once the basic structure of processes is given, the details of their particular relationships should be analysed in order to harmonise the cooperation with the internal structure and contents of each process. Structural harmony means the synchronisation of the internal process run with the run of the other processes – partners in the cooperation. Content harmony means taking each cooperation point as an act of communication of both processes. Considering this cooperation point as a service one can think about both dimensions in a harmonious whole: service always means delivering the right product at the right time³.

Detailed analysis of the business process cooperation naturally brings the consequential questions:

- Why and how should processes in the organisation cooperate?
- Which attributes should their cooperation have (time, quality)?

² Central European Association for Business Process Management (see <http://www.ceabpm.eu>)

³ An example of disharmony in business processes can be seen in the situation, still not uncommon in big companies, when customers are waiting a substantial time for a response and begins their own investigation into the reasons. They usually run through a number of the organisation's units mapping the natural but unmanaged 'business process' to find out finally that there is some deadlock in the organisation which may in their case mean the process can never end. The deadlock is usually caused by inconsistency in the defined competences and responsibilities of different roles in the process, usually as a consequence of the fact that competences are defined without respect to possible specific situations. As the process attendees are not regarded as process roles but rather as members of organisation units in a traditional organisation, it is almost impossible to view the whole process and all its possible exceptions and their consequences even in detailed competences. To reliably avoid such a situation it is necessary to recognise all points of necessary communication in the process, which is impossible from the point of view of the hierarchical organisation structure.

As is argued above, the cooperation of processes always means their communication. The need for the cooperation follows primarily from the mutual positions of both processes. According to the above mentioned MMABP methodology and consistently with the ideas of process based management there are only two correct reasons for the existence of the process:

The purpose of the key process is implicitly undoubted: it is given by the fact that this process represents the direct way of satisfying the need of a customer, which is the universal mission of any organisation. The key process always represents the direct service to customers;

The purpose of the supporting process is given by the services by which this process is supporting other processes.

Any cooperation between processes always means providing the service either directly for the customer or indirectly by supporting other processes. MMABP methodology contains the technique for design of the cooperation structure of processes via “internal outsourcing” of producing process chains from the key processes. This way the natural supporting processes are identified and cooperation, as well as the basis of the structure of processes in the organisation, is established.

Process-oriented infrastructure

To utilise the business process system in real life it is necessary to create the required infrastructures. There are two main kinds of infrastructures representing the two main resources in an organisation: technological infrastructure representing the aspects of automation, and organisational infrastructure representing the human aspects of the organisation's behaviour. As the main goal of the process based management of the organisation is to make it principally and permanently flexible, its infrastructures also need to have these attributes. Thus there are two crucial questions to answer regarding the implementation of the process-managed organisation:

- How to organise a permanently flexible organisation?
- How to design the information system of a permanently flexible organisation?

In [35] the methodology for the design of a process based organisation is presented. The last step in the procedure called “Building resulting infrastructures” is based on the work with the structure of services identified in previous steps. Services are identified as a general meaning of the relationships among business processes – their mutual cooperation. Details of every service (alias cooperation act) are described in the form of a Service Level Agreement (SLA) and are used in the last step of the procedure as a common basis for creation of all required infrastructures: organisation and information systems. The organisation structure of the organisation is then built directly on the structure of competencies derived from the mutual competency relationships of processes that are defined in their common SLA. So the rights and responsibilities of managers, as well as regular attendees of both processes, are directly following from the requirements of the processes. This way the organisation structure is flexible and exactly in accordance with the flexibility of the processes.

Similarly the structure of the information system is derived from the mutual relationships of processes that are defined in their common SLA. The SLA defines all necessary products of the service (alias processes cooperation act) and their quality, as well as time attributes, which is a perfectly sufficient basis for deciding on the necessary functionality of particular parts of the information system. The particular behaviour of the system is then given by the process itself, and because of the basic functionality is principally recognised as an integral part of the system by the workflow-management engine. The workflow management system is thus the basic condition for making the information system of the organisation flexible enough in terms of the main principle of process based organisation.

The common intersection of all three viewpoints is characterised by the concept of Service, which represents their universal common meaning. The concept of **Service**, as it is discussed above, from all three viewpoints represents a **common denominator** for the content, technical, and human aspects of organisation management.

Chapter 2. Business systems modelling

This chapter deals with the “*Business System Model*” concept. In the MMABP methodology, we understand by this the semi-formal model of a business system consisting of business objects and business processes. This conception is slightly different from the usual meaning of the “*Business Model*” concept, which usually means some stated intention articulating the logic, data, and other evidence that support a value proposition for the customer, and a viable structure of revenues and costs for the enterprise delivering that value. The “*model*” concept is thereby close to being a synonym of conception or draft. In our approach the “*model*” is a more formal issue in the form of diagrams and completing formal definitions of their elements. It is important to distinguish between these two different meanings of the common term “*model*”.

The crucial basic principle of Information System (IS) development is the **Principle of Modelling**.

This principle expresses the presumption that the *objective basis for the implementation of the business system in the organisation must be constituted by real facts existing outside of, and independently of, the organisation*. The real facts that are regarded as relevant are those which substantially influence the possibility of the organisation to achieve its objectives. These facts are visible in the form of specific (critical) values of so-called critical factors. Each real world object playing any important role in the business system can be modelled as a collection of attributes expressing those critical factors. We call them **business objects**. Critical changes in the critical factor values are recognised as (external) events. Events are regarded here as the only reason to start the **business process** – process trigger⁴.

In the area of business processes, the principle of modelling states that the system of business processes in the organisation is the model of relationships between objectives and critical events, and mutual relationships between the objectives and between the events. The purpose of each business process in the organisation is to ensure the proper reaction for any given event. Essential relationships between an organisation’s objectives, critical factors and events are expressed in the form of relationships between particular processes. Products of those processes as well as their actors, goals, problems, circumstances and other aspects should correspond to the relevant business objects.

The purpose of the principle of modelling in the area of business modelling is:

⁴ The concept of events is very wide here – it even covers such changes of facts which are not usually regarded as “changes of critical factors values”. For example, customer requests, or changes of production technology parameters are also regarded here as events (i.e. “critical” changes).

- it defines the basis for the analysis (the **essential substance** to be analysed)
- it leads to creation of a **system of business processes** which:
 - is able to react to each significant change that also requires a change in business processes (changes of goals, objectives and critical factors)
 - is optimal – consisting of all processes, and only those which are necessary under given business conditions.

Information System, as an infrastructure for the business system, has to be based on the same objective model of the real world. Such an objective model of the real world is traditionally called the **conceptual model**.

The “conceptual” concept was first used in the area of data modelling. It expresses the fact that the database should describe the essential characteristics of the real world: **objects and their mutual relationships**.

There are three mutually dependent main principles, which together explain the sense and the reason for the term “conceptual”:

- The Principle of Modelling,
- The Principle of Three Architectures, and
- The Principle of Abstraction.

From the data point of view, the contents and the structure of database objects reflect the contents and structure of the real world objects. The correctness of the data model is measured via its similarity to the real world. In order to make such measurements, the term “similarity” must be exactly defined. Therefore, the special Entity Relationship Diagram (ERD) has been developed ([8]). ERD describes the essential structural characteristics of the real world: objects and their mutual relationships. It is constructed to be able to exactly describe the objects and their relationships in the same way as we see them in the real world. At the same time, this model describes the essential requirements for the database – it must contain the information about the same objects and their relationships. The form in which the particular database describes these facts always depends on the technological and implementation characteristics of the environment in which the database is realised. However, the essential shape of the model still remains the same. Because of the need to describe the same database in its various shapes (essential, technological, implemented), the principle of different architectures has been formulated. This principle, generalised to the scope of the whole system (not only its database), is called “**The Principle of Three Architectures**” (see [27]).

The Principle of the Modelling also proves to be too general – some parts of the system processes have to be regarded as the model of the real world. Although the main problem of the so-called structured approach to the IS development is that it is not able to recognise which system processes form the model of the real world and which do not. Such recognition requires separation of the modelling operations from the others and organising them into the special algorithms according to real world objects and their relationships. And this point of view is not reachable under the “structured paradigm” without accepting the natural unity of the modelling system processes and the data in the database. Acceptance of the natural unity of the modelling processes and the data entities formulated as the main OO principle enables us to solve Yourdon’s problems ([46]) with control processes – the essential controlling algorithms follow on from the entity life histories.

As shown above, the Modelling Principle seems to be general and independent of existing paradigms. Each new paradigm can only specify its place in IS development but cannot eliminate or limit it.

The Principle of Abstraction expresses the fatal need for creating abstract concepts while modelling the real world. There are many possible classifications of abstractions. Undoubtedly, **hierarchical abstractions** are very important kinds of abstraction.

Hierarchical abstractions are the means for breaking down the elements of the designed Information System to the level of detail. Higher level concepts consist of the lower level ones. On each level of detail the elements of the developed IS and their relationships are described. The elements on each higher (i.e. non-elementary) level of detail are abstract concepts. Only the lowest (i.e. most detailed, elementary) level contains definite elements. There is the “tree structure” of dependencies between the concepts of the higher and lower levels. This is so that each element has one parent element on the higher level (with the exception of the highest element – the root of the tree) and can have several child elements on the lower level (with the exception of the lowest elements – the leaves of the tree). Hierarchical abstractions are of two basic types:

- **Aggregation.** Subordinated elements are parts of the superior concept.
- **Generalisation.** Subordinated elements are particular types of the superior concept.

The aggregation type of abstraction is typically used for breaking down processes – functions into sub-functions (using the Top-Down procedure), while the generalisation type of abstraction is typically used for breaking down conceptual objects into sub-objects (specialisation into object types). The incompatibility of these two basic approaches with the concept break-down forms the basis for a lot of problems and misunderstandings, not only in structured methods (it has often been a source of vital problems for the “structured paradigm”), but also in object-oriented methods.

All three of the main above-mentioned principles are present in all particular aspects of the given methodology and are therefore are casually revisited in different contexts in following chapters.

2.1 Conceptual Modelling under the object paradigm

To successfully manage the effectiveness as well as the prospective development of an organisation it is necessary to precisely know all of the important aspects of its contents together with their mutual relationships. Informatics offers a semi-formal technique called “conceptual modelling” for this purpose. It has been developed in the field of data management as a sub-discipline of information systems development. From the certain level of complexity of the real world system (usually called a “business system”), which is described by data handled in the information system, there must be a precise systematic view which ensures that the real world system is truthfully and completely described with the data in the information system. The conceptual model fulfils this requirement expressing the real world as a system of objects and their relationships and defining how to represent it by data in the information system.

As previously mentioned, the “conceptual” concept was first used in the area of data modelling. This origin is still visible in the common understanding of the adjective “conceptual” in the sense of modelling with the standard tool for object-oriented modelling – the Unified Modelling Language (UML – see [23]):

Object-oriented analysis and design materials written by Craig Larman for ObjectSpace (www.objectspace.com) describe the **conceptual modelling** as follows:

- Classes representing concepts from the real-world domain.
- Binary associations describe relationships between two concepts.
- The concepts can have attributes but no operations.
- General associations indicate that the specialised concepts are subsets of a more general concept. The specialised concepts have associations or attributes that are not in the general concept.
- Each association conclusion can have graphical adornments indicating their end name, multiplicity, and much more.

Cris Kobryn ([13]), Co-Chair of the UML Revision Task Force, takes the conceptual model into account when describing the “Structural Model” as a view **of a system that emphasises the structure of objects, including their classifiers, relationships, attributes and operations**. The purpose of such a model is to show the static structure of the system:

- the entities that exist,
- internal structure,
- relationship to other entities.

In addition, Kobryn gives several tips for structural modelling:

- Define a “skeleton” (or “backbone”) that can be extended and refined as you learn more about your domain.
- Focus on using basic constructs well; add advanced constructs and/or notations only as required.
- Defer implementation concerns until later on in the modelling process.
- Structural diagrams should:
 - emphasise a particular aspect of the structural model
 - contain classifiers at the same level of abstraction.

- Large numbers of classifiers should be organised into packages.

Roni Weisman ([45]) from Softera also elaborates on the “Conceptual System Model”. He distinguishes three types of objects:

- Entity (object which hold the system’s data),
- Boundary Object (interface objects which directly interact with the external world – actors),
- Control Object (objects which manage the system operations).

As may be seen from previous paragraphs, there are several approaches to conceptual modelling in the area of object-oriented methods. Each of them reduces the Object Model (represented by the Class Diagram) to the model of objects and the relationships between them, represented by their attributes, but not by their methods. This reduction is also presented in Roni Weisman’s approach (see above), even if he takes “Entities”, as well as “Control Object” into consideration. Just the fact of distinguishing between “static” and “dynamic ensuring” objects is the best demonstration of such a reduction. The common understanding of the term “conceptual” thus tends towards the synonym for “static”.

This chapter argues for the really “object-oriented” approach to conceptual modelling. At first it means that it is necessary to not only model static aspects of the real world, but also its dynamics. The existence of the object as the collection of data (attributes) and functions (methods) is to be the right reason for data processing operations control. Moreover, regarding the conceptual point of view together with the principles of object orientation, one must stop taking the object’s methods simply as a collection of procedures usable for communication with other objects. One should seek the wider, conceptual sense of them as a whole – seek the substance of their synergy. Such a “higher sense” of the object’s methods represents the **object life cycle**.

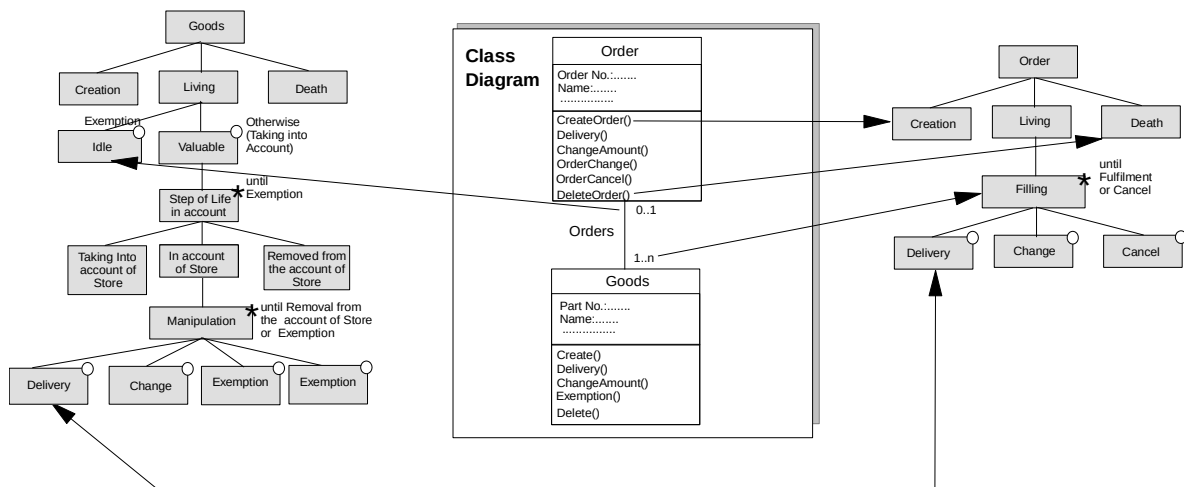


Figure 4: Object life cycles versus object model using Jackson's Structure Diagrams, Source: Author

Figure 4 illustrates the object life cycle as a complement to the Class Diagram. It can easily be seen that all methods of the conceptual object should be ordered into one algorithm that describes each method's place in the overall process of the object's life. This place defines the conceptual sense of the method. In this sense it is obviously absurd to consider such methods as "give_list" or "send_status", as well as being absurd to speak of "sending messages" between objects (human-like discussion between the Order and the Goods in this example). Such a point of view is suitable for the model of objects in a program system, but in the case of conceptual objects it is clearly improper.

The figure also indicates the fact that there are dependencies between the methods of the different objects which correspond to the association between objects not only in the sense of existence (method "Delivery" in this example), but also in the "structural sense" – in the sense of the structure of a life cycle. So in this example, the fact that "Goods do not need to be ordered" (see the partiality of the association between Goods and Order) corresponds to the possibility of the "idle living" of Goods. Similarly, the fact that "Goods may be reordered" (see the cardinality \underline{n} of the association) corresponds to the cycle of the "filling" part of the life of the Order. In order to better understand structural dependencies in general, which itself is the best way to precisely define those object-oriented conceptual modelling principles, see Jackson's JSD [10].

it seems that with the topic of conceptual modelling, the "renaissance" of clearly correct and unambiguous general principles of structured programming according to Jackson could come⁵.

⁵ The influence of Jackson's theory is discussed in the chapter 5.2 *Structural coherency* as a topic of "structural consistency".

2.2 Two times two basic dimensions of the Real World business model

According to the principle of modelling, the model of the information system must be based upon the model of the real world. By “real world” we mean the objective substance of the activities to be supported by the IS and of the facts to be stored in the IS. This demand is only met in the “static” parts of the conceptual model in the traditional sense (i.e. in the data or object model of the reality). In the system behaviour model (behavioural UML diagrams, Use Cases, etc.) we model the Information System’s dynamics rather than the dynamics of the real world. We not only model the objects, but also the users, of the IS; not only information sources but also its targets. On the other hand, it is also clear that the way in which the IS should behave (and should be used) is substantial. It arises from the rules of the real world – from business activities which define the sense of the IS in the form of the business need for information. So the crucial question is: what are the substantial real world actions and processes to be modelled?

Some solution is offered by the object model (class diagram) itself. The model of the real world, as the system of objects encapsulating the data with appropriate actions, not only speaks about the data which the IS stores, but also about the actions with the data and their sequences (processes). The system of conceptual objects, and their interaction, models the part of the real world dynamics that follows from the nature of the objects (their life cycles) and their relationships. However, it does not model that part of the real world dynamics which follows from the substance of the information need – from the nature of the business.

So, there are two kinds of “dynamics” of the real world to be analysed within the process of IS development:

- Dynamics of the real world objects and their relationships given by their conceptual nature (real world conditions and constraints),
- Dynamics of the business activities given by the conceptual nature of business processes (business nature).

We may conclude from the previous paragraph that there are two basic orthogonal views of the “real world”:

- The object view which emphasises the **substance of the real world**,
- The process view that emphasises the **real world behaviour** (see Figure 5).

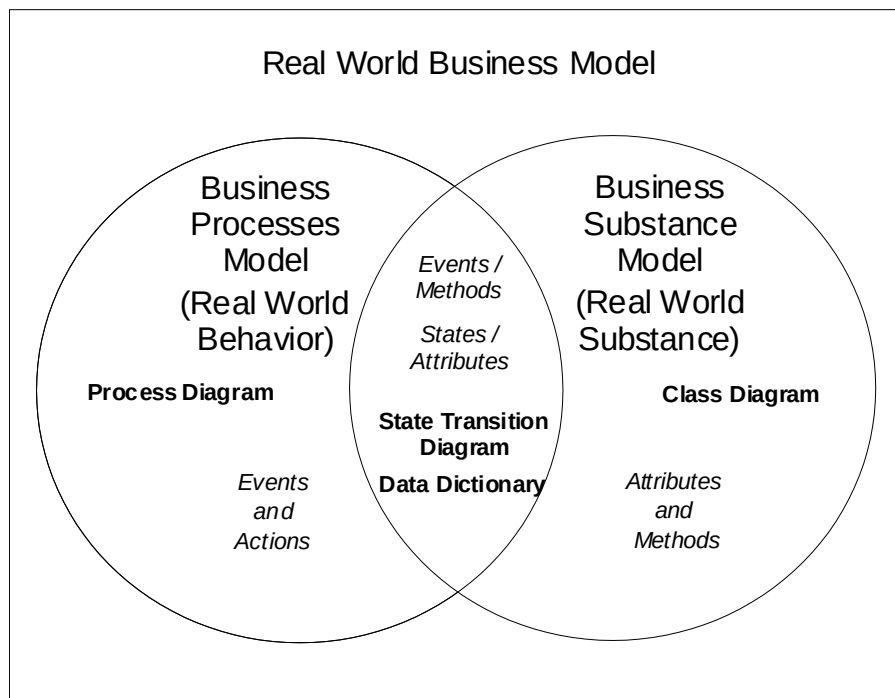


Figure 5: Two parts of the Real World Business Model

Source: Author

The first view (Business Substance Model) represents **objects and their mutual relationships** consisting of attributes and methods.

The second view (Business Process Model) represents **business processes** consisting of events and actions.

Of course, the model of objects also concerns their behaviour in the form of entity life algorithms (method ordering). Such behaviour is seen from the point of view of objects and their relationships. It says nothing about the superior reasons for it. Thus, the **behaviour of the objects should be regarded as the structural aspect of the real world**⁶.

⁶ By the term “structural aspect” we mean that, although we speak about the behaviour, we do not describe some process aimed on meeting some goal (like in the case of “business process”). The description of the life cycle is rather a complement to the Class model defining in detail the objective rules for ordering operations provided by or on the given object (what precedes what under which conditions and circumstances). Such description is just an algorithmic definition of real-world rules instead of a description of some intentional “process”.

The significant aspect of the real world's behaviour, seen from the process point of view, which is not present in the object point of view, is that there has to be a superior reason for the real world behaviour, independent of the object life rules. In practice, it means that for each business process, some reason in the form of the goal, objective, and/or external input event (customer requirement) must exist. The business process as the collection of the actions, ordered by time, and influencing the objects (their internal states and their mutual behaviour), is something more than just an amorphous collection of actions.

Similar to the structural modelling of the real world (Business Substance Model), even for the behavioural model (Process Model), it is necessary to describe the principles and general rules, and develop the tool (diagrammatic technique) which reflects them. [28] roughly states the basic principles and general rules of process modelling and also contains the basic specification of the necessary diagrammatic tool – the Process Diagram – in the form of the Business Process Meta-Model. The following examples use the notation of that Process Diagram.

Figure 5 illustrates the idea of the two basic dimensions of the business system model:

- Structure (substance) of the Real World (the view on the Real World as a set of objects and their relationships),
- Behaviour of the Real World (the view on the Real World as a set of mutually connected business processes).

Both dimensions have a common intersection which contains, besides the static object aspects as attributes and data structures, also typical dynamic aspects as events, methods, and object states. Thus the description of dynamics is not just the matter of the behavioural model, but is also the matter of the conceptual model.

Business process is a process of achievement of the human will. It has the goal, and the product(s). It typically combines different business objects giving them specific meaning (roles of actors, products, etc.). Business objects may be specified in detail by the description of their life cycles. An object life cycle is a description of business rules connected with the object in terms of states and transitions. Objects are typically taking different roles in different processes giving them the context (Real World rules), while the business process, following the process goal, typically connects the lives of several objects. For a detailed discussion of the main differences between object life cycles and business processes see [33], [34].

Besides the two-dimensional approach to the business system modelling there are also two basic complementary views on the system (see Figure 6):

- global view on the whole system abstracting from details,

- detailed view on just a part of the system abstracting from the whole.

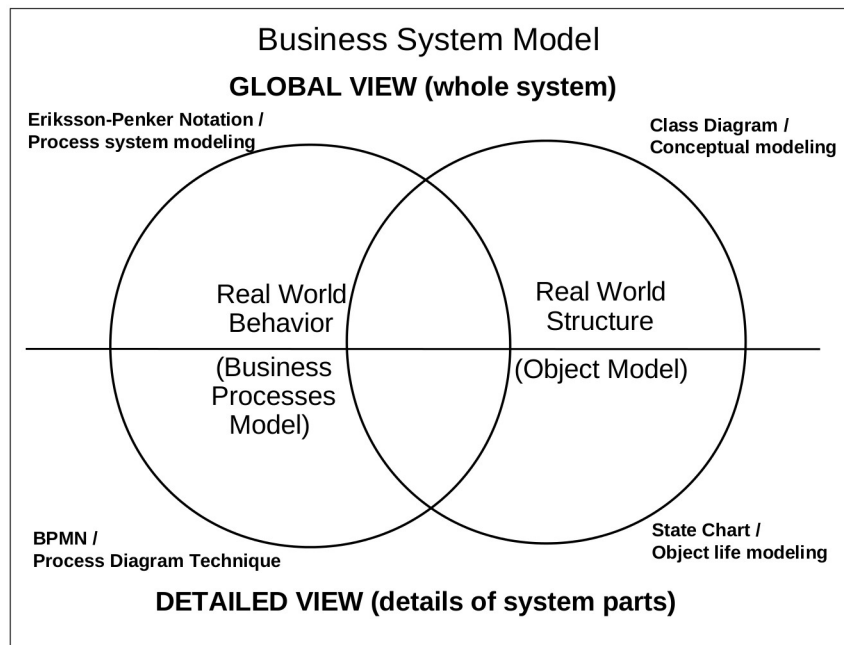


Figure 6: Two views on two dimensions of a business system

Source: Author

Each global model describes the structure (objects in general) of the system, while each detailed model is oriented on the dynamics (processes in general). Nevertheless it does not void the previous principle of two basic dimensions of the Real World Model. If we combine these two basic dimensions with two basic views we get four basic kinds of the business system model:

- global model of objects – conceptual model (Class Diagram)
- detailed model of one object – object life cycle (State Chart)
- global model of processes – model of the system of processes (Eriksson-Penker Diagram)
- detailed model of one process – model of the process run (BPMN diagram)

So we can speak about the process view on an object (detailed description of object's life cycle) as well as about the object view on business processes (global process model). In other words the world of objects also has its dynamics (behaviour), and the world of processes also has its structure (objects).

view/subject	Business Objects	Business Processes
Global view	conceptual model (Class Diagram)	process system model (Eriksson-Penker Diagram / TOGAF Event Diagram)
Detailed view	object life cycle (State Chart)	model of the process flow (BPMN diagram)

Table 1. Two basic views on two basic subjects of interest in a business system

Table 1 shows how the *BPMN* language as a process-oriented description of the business process covers only one part of the behavioural dimension of the Business System Model. *BPMN* does not cover the problem of global modelling of the process system. The most accepted modelling standard suitable for this purpose is the *Eriksson-Penker Notation* ([6]) created as an extension of the *UML*, which corresponds well with the fact that the global view on processes is object-oriented – it is the conceptual model of business processes in fact. Currently, the most suitable diagram for the system model of processes (process map) is the Event Diagram from the standard TOGAF ([44]), which nevertheless, is also based on the original Eriksson-Penker notation.

On the other hand both Business Object dimensions are fully covered by the *UML* with its two basic diagrams: *Class Diagram* and *State Chart Diagram*.

All the languages mentioned above are explained in detail in following chapters.

2.3 Business Modelling Specification

MMABP methodology defines the above mentioned basic principles, elements, and relationships in a Business System Model as a semi-formal definition using the UML Class Diagram. The so-called “Business Modelling Specification” (see Figure 7) consists of three associated packages:

- Business Substance meta-model package,
- Business Process meta-model package,
- Business Models Consistency package.

Business substance and business processes represent two basic dimensions of the real world model mentioned in the previous text. Each of the two packages specifies the basic concepts required for a model of a given dimension together with the basic rules for expressing the business logic given by the dimension. As they are modelling the general basis of all possible models in a given dimension, they are both meta-models.

Unlike the other two packages, the Business Models Consistency package is not a meta-model to all intents and purposes. It models the general basis of the mutual interconnections and dependencies of both meta-models. In that sense, it extends both meta-models with new concepts in order to address the general mutual dependencies of the real world models.

The business substance model is based on the UML Class Model with minimal extensions. The business process model has its own rules that are not present in the current version of the UML. The consistency rules for both models are not present in the current version of the UML, either.

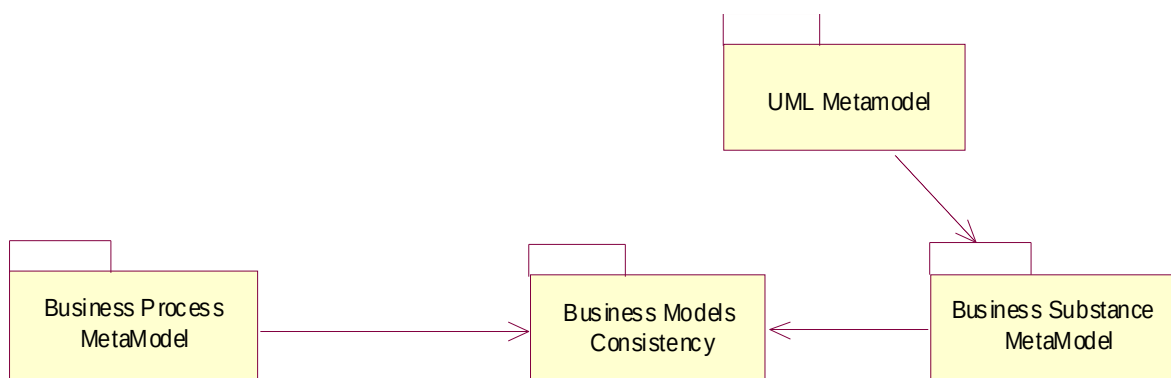


Figure 7: Business Modelling Specification overview

Source: Author

Business Substance Meta-model

The **Business Substance Meta-model package** (see Figure 8) specifies the basic concepts required for a model of a business substance and defines the basic needs/possibilities of their mutual interconnections (i.e. business substance modelling logic – “how to model what the real world is”).

it is based on the Core Foundation Package of the UML meta-model (see [23]), which it reduces, as well as extends, for the purpose of business substance modelling.

Through this meta-model, the Core Foundation Package of the UML meta-model is reduced to the concepts and constructions that are relevant to the purpose of modelling the business substance. Such models are usually called “conceptual”. Unfortunately, the term “conceptual” is closely connected with the term “static”, which we regard as an improper reduction for the model of the real world, which is naturally dynamic (see previous sections).

Therefore, this meta-model also extends to the UML meta-model with the following concepts:

- The Class State as a subtype of the Behavioural Feature complements the concept Method. The purpose of its existence is to allow the taking of all the methods of a given class as an ordered whole in the time dimension. Such a chronologically ordered aggregate of actions is usually called an “algorithm”. The algorithm structure of all methods of a given class is called the Class Life Cycle.

The Class Life Cycle is the abstract name for the role of “Class” as an aggregate of the Class Life Cycle Steps. The Class Life Cycle Step is a collection of:

- a possible one input state,
- at least one output state,
- possibly more processed attributes, and
- one processing method.

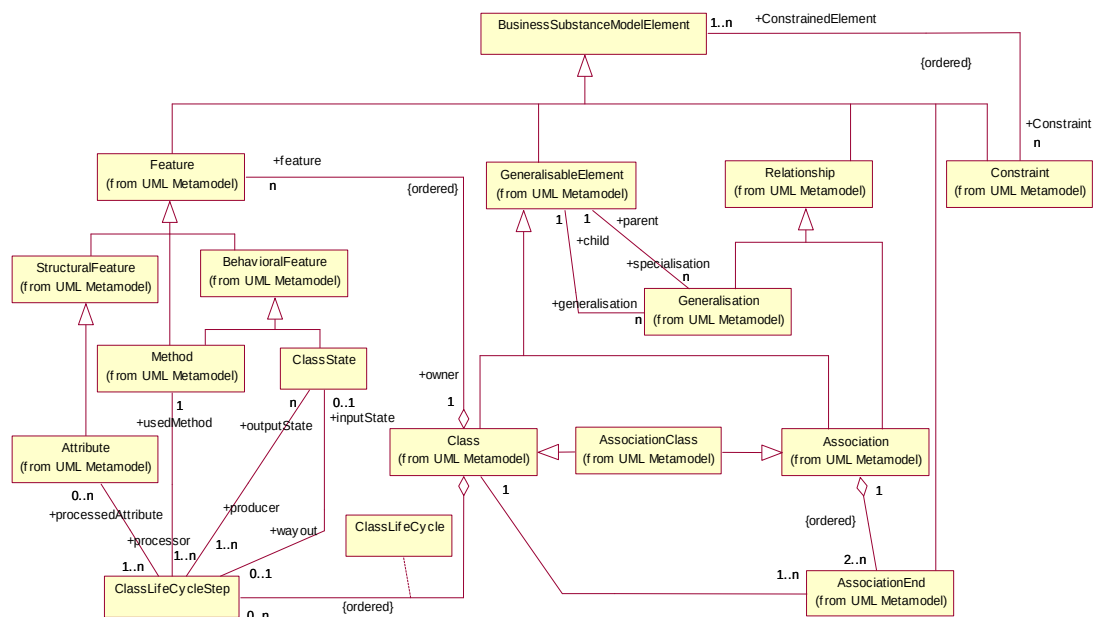


Figure 8: Business Substance meta-model package

Source: Author

As “Class” is a generalizable element, so too should the Class Life Cycle and the Class Life Cycle Step inherit this feature. In practice, it requires the specification of general rules for generalising algorithms, especially in the context of the consistency rules for both business models. In the current state of our work, this topic has not yet been elaborated upon. We can find some inspiration in the work of M.A. Jackson, as was mentioned above ([9], [10]).

Business Process Meta-model

The Business Process meta-model package (see Figure 9) specifies the basic concepts required for a business process model and defines the basic needs/possibilities of their mutual interconnections (i.e. business process modelling logic – “how to model; how the real world behaves”).

- Control Activity as a Stimulus inherits the stimulation competence. Together with this fact it follows on from the multiplicity 1 (i.e. monopoly) of the stimulation association that the Processing Activity can be stimulated either by the Event or by the Control Activity exclusively.
- Each Stimulus has to have at least one input state except for the very first Stimulus (Terminal Event), which has no input state. This exception is expressed by the specific zero-multiplicity association with the Terminal Event, which overwrites the inherited general association between the Stimulus and the Non-Terminal State.
- Each State has to be an input for at least one Stimulus except the very last one (the Terminal State), which has no succeeding activity.
- The Characteristics of the terminal event, as well as of the terminal state, are relative to the specified model. Usage of the model as a part (sub-process) of another model will change all terminal events of the sub-process to regular ones and all terminal states to internal ones from the super-process point of view.
- The Processing Activity as well as the Decision is a composite aggregate of the input/Output Sets. As follows from this fact, one particular input/Output Set can input exclusively to either the Decision or the Processing Activity.

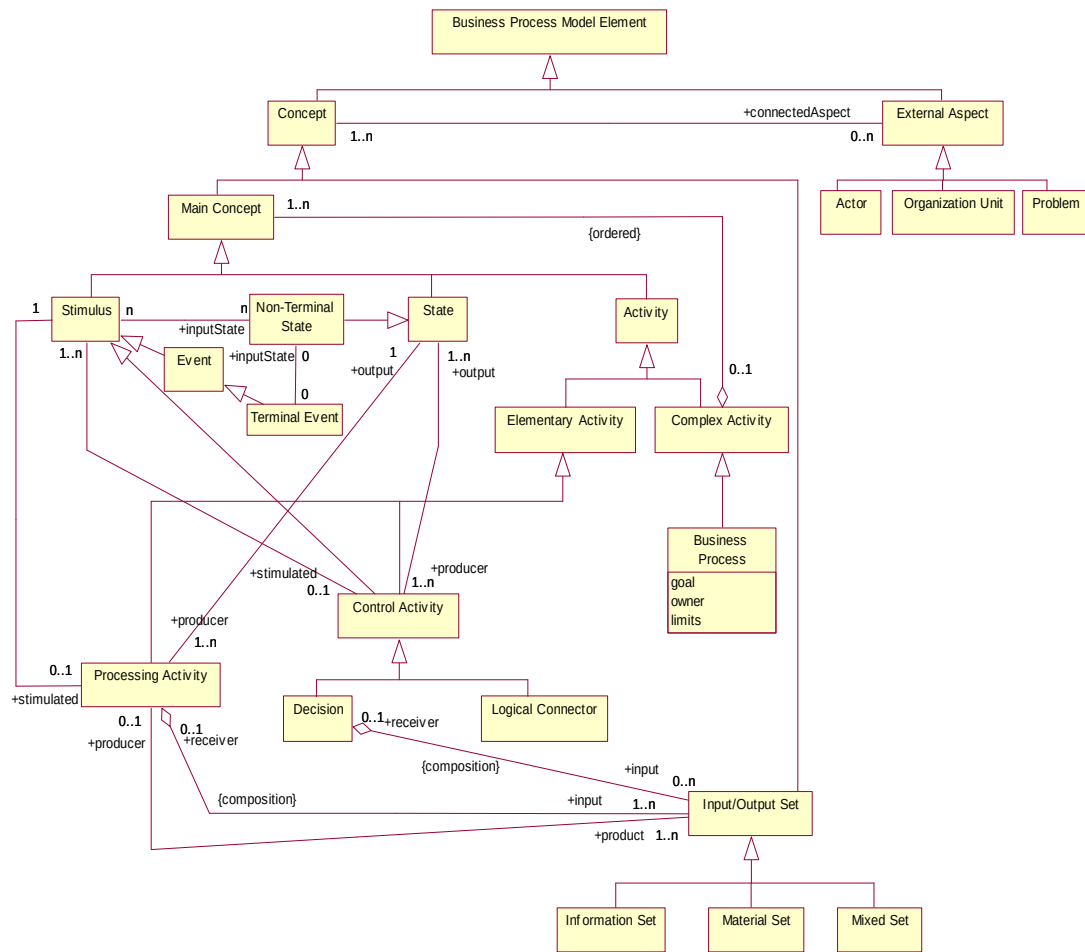


Figure 9: Business Process Meta-model package
 Source: Author

- To ensure “structural consistency” it is necessary to describe the class life cycle “in a structural manner”. Therefore, this model extends the Business Substance meta-model with the new concepts of: Structure, Structure Element, Structural Consistency Element, and Association Ends Couple.
- Each Class Life Cycle Step is a Structure Element, and as such, it is a part of the Structure. Structure is an ordered aggregation of Structure elements where each Structure Element is either a Structure Leaf or a Structure (as an Abstract Element) exclusively. Each Structure Element can, but need not be the Class Life Cycle Step, except for a Structure Leaf that always must be in the Class Life Cycle Step. In the model, this is expressed by the non-optional association “ISA” between the Structure Leaf and the Class Life Cycle Step that overwrites the optional general association between the Class Life Cycle Step and the Structure Element.
- Structure must, and can only be one of the three types: Sequence, Selection, and iteration.
- The abstract “Opposite Class Multiplicity” concept addresses the opposite end of each association of a given Class. This is necessary in order to associate it with the proper Structure type (iteration/Sequence/Selection) of the Structure Element describing the part of the Class Life Cycle.

Such a connection allows the expression of the structural correspondence between the association opposite end multiplicity and the appropriate type of life cycle structure element:

the multiplicity of each association opposite end requires at least one iteration in the class life cycle of a given Class,

the monopoly of each association opposite end requires at least one sequence in the class life cycle of a given Class, and

the optionality of each association opposite end requires at least one selection in the class life cycle of a given Class.

This chapter describes the systemic approach to the modelling of business systems based on the formal business meta-model. It arises from the belief that the main general principles underlying the idea of conceptual modelling (i.e. modelling, abstraction, and different architectures) are the best validation of such features of a modelling language. MMABP defines the set of (meta)models that are intended to be systematically and continuously complemented with the set of rules and other tools reflecting the main practical problems and challenges, many of them discussed above, for example:

- Completeness of the business process modelling rules.

- Completeness of the business substance modelling rules.
- Completeness of the consistency rules in the Models Consistency Model.
- Problem of life cycle generalisation.

For the practical view of the topic of consistency as well as some examples, see *Chapter 5 Consistency of business system models*.

Chapter 3. Modelling business processes

In this chapter, we explain how to model business processes in the organization with a full respect to the basic principles of the process-driven management on one hand, and to the technical substance and the essential role of information technology in the process-driven management on the other hand.

The respect to the basic principles of the process-driven management requires to permanently keep in mind the real meaning and purpose of business processes and process-driven view of the business, and consequently, to avoid the popular overestimating of the technical aspects that often leads to the self-purpose modelling of processes out of their business meaning and purpose or even contradicting with them.

The respect to the technical substance of business processes and the essential role of information technology in process-driven management requires to model the business processes so precisely and exactly to be able to fully exploit the opportunities of the technology development in the development of the business system.

In terms of the above-described aspects, we introduce the MMABP view of how to create the Process Map and related detailed models of selected business processes. MMABP uses the standard process modelling languages BPMN and TOGAF Event Diagram. Nevertheless, the process modelling languages are discussed in a broader context in this chapter. A special attention is paid to the important aspects of the business process modelling: intentionality and its consequences in the detailed process modelling technique, process memory, and MMABP abstraction levels of process models.

3.1 Modelling the system of business processes (Process Map)

We understand the term “system of business processes” to mean the global view of processes. This model expresses which processes in which mutual relationships form the business system. The system view of processes is principally object-oriented, which means that it is focused on processes as objects and can simply recognise their existence and mutual context, not their dynamic details. This means that although this model describes processes, it represents a static view of them and cannot describe their dynamic aspects, for which the description of the detailed process model is intended.

One of the most accepted “de facto” standards which fully supports the system (object-oriented) view of business processes is the Eriksson-Penker Notation ([6]). It was created as an extension of the UML ([23]), which corresponds with the above discussed “object nature” of the global view on processes. This notation distinguishes between the “Business Process View”, which illustrates the interaction between different processes, and the “Business Behavioural View”, which describes the individual behaviour of the particular process. In this way it respects the important difference between the global object-oriented view of a process system and the detailed process-oriented view of a single process. The Eriksson-Penker process diagram, usually called a **Process Map** became a commonly accepted standard as a complement to the BPMN that compensates the absence of the global view in this language. Also the traditional ARIS methodology ([41]) respects the need to distinguish between the global (object-oriented) view of the system of processes and the detailed (process-oriented) view of a single process. ARIS uses so-called VAC (Value Added Chain) notation for this purpose. The name “VAC” points to the fact that ARIS methodology emphasises some specific aspects (“adding the value”) that are not fully relevant to the general meaning of the global process model. Nevertheless, the ARIS methodology this way allows viewing all processes as a system, which is undoubtedly a significant quality of this approach.

Nowadays, the most perspective standard for the system view of processes is the **TOGAF Event Diagram**. It is a part of The Open Group Architecture Framework (TOGAF [44]), the most popular methodological framework in the field of Enterprise Architecture. As a part of the enterprise architecture framework, this diagram describes not only the system of processes (using the form of the Eriksson-Penker process diagram) but also their relationships to the general (functional) structure of an organization. The MMABP methodology, therefore, uses the TOGAF Event Diagram for the description of the enterprise system at two different levels of abstraction: Functional Structure of the enterprise that covers several relatively autonomous process systems, and Process Map expressing an internal structure of one process system. For further explanation of the methodical need for describing the processes on different abstraction levels see *Chapter 3.3 Process memory and MMABP process abstraction levels*.

Figure 11 shows an example of the system of processes in the TOGAF Event Diagram. The key process *Client Management* represents the value that the organisation delivers to its customers. In this way, this process is directly connected with the strategy of the organization. The organization provides the customer with various, mutually related services during the whole customer's "life cycle", i.e. the period of the collaboration of the organization with the customer. It begins with the acquisition of the customer and ends with the termination of the communication with the customer. Other processes in the model serve as supporting processes. Their value in the organisation is given by the provided support instead of the direct relation to some strategic goal. Each relationship between two processes represents the particular service provided by the supporting process to the supported one. The *Contracting Management* process represents the activities of creating the contract with the customer. The key process needs this service when the customer orders the particular service. The *Service Providing* process covers the whole period of provision of the customer with one service. The *Complaint Management* process represents the handling of the customer's perspective complaint. The key process consumes the support processes' services repeatedly to provide the customer with the chain of mutually related different services.

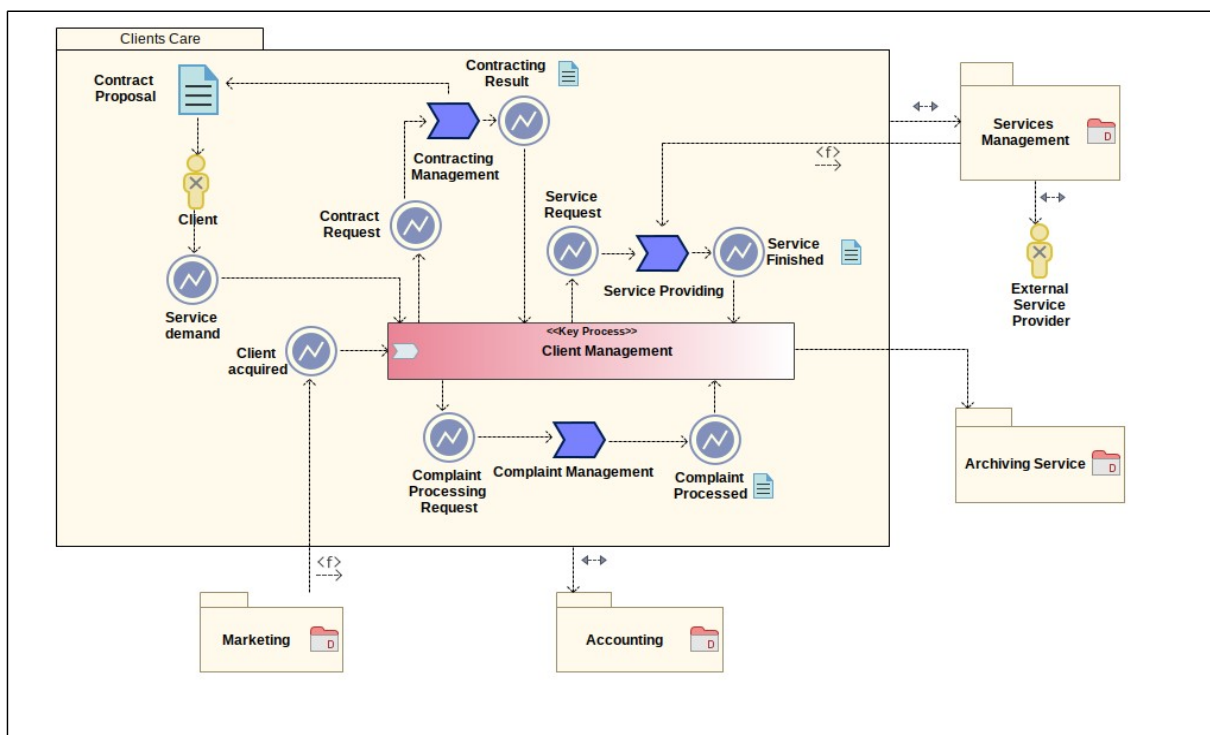


Figure 11: Global process model (Process Map) in the TOGAF Event Diagram

Source: Author

The example at Figure 11 also shows the functional structure of the organization. Particular functional domains are described as “UML packages”. There are five functional domains in the example: Clients Care (the domain of the enterprise’s key function), and support domains Services Management, Archiving Service, Marketing, and Accounting. Every domain can be regarded as a relatively autonomous system of business processes, where one can see also the relative, local “key” process of the domain and other processes as relatively supporting processes of the domain. In this example, only the Clients Care domain is decomposed to the system of processes, the processes of other domains are not visible. Event Diagram can show the collaborative relationships between particular processes in the domain as well as across different domains, the relationships between the process and the domain, and also the most general relationships between the domains.

3.2 Modelling business process details

A significant aspect of real world behaviour seen from the process point of view, which is not present in the object point of view, is that there must be a superior reason for real world behaviour, independent of the object life rules. In practice, this means that for each business process, some reason in the form of the goal, objective, and/or external input event (customer requirement) must exist. Business process, as the collection of the actions, ordered by time and influencing the objects (their internal states and their mutual behaviour), is something more than just a random collection of actions.

Similarly to the structural modelling of the real world (Object Model), even for the behavioural model (Process Model), it is necessary to describe the principles and general rules, and develop the tool (diagrammatic technique) which reflects them. The **Process Diagram**, presented in the following text, is a general idea of essential elements and attributes of the detailed process model developed as a part of the MMABP Methodology (see [27]); see also the MMABP Business Process meta-model presented in *Chapter 2. Business systems modelling*).

The methodology is focused on creating the model of the system of business processes which:

- respects the basic objectives and goals, current state and specific characteristics of the organisation,
- respects the objective circumstances (those which arise outside the organisation and are independent of the organisation), which can play a significant role in the behaviour of the organisation,
- is “optimal” in the sense of the economic efficiency of the processes,
- is “optimal” in the sense of the maximum simplicity, together with total functionality,
- is prepared for later optimisation, implementation and installation of the system of processes which respect the characteristics described above.

The idea of the **Process Diagram** is also influenced by some ideas of information Analysis (see [15]).

For the purpose of modelling the real world processes, we use the term **Conceptual Business Processes Model**⁸. The Conceptual Business Processes Model describes those processes that are necessary for achieving the business goals of the organisation, and thus, are to be implemented as the workflow and supported by the Information System (IS). Such business processes are not influenced by the information technology aspects and work as a common basis for IS development, together with workflow implementation and business processes re-engineering.

The purpose of the Process Diagram as a main tool for business process modelling is to express the basic general regularity and rules of the real world in terms of a business process. The technique has to be independent of concrete conditions of the process implementation (business technology, organisation, etc.) as well as conditions of the process instance and supporting information technology (workflow definition, database workflow support).

With respect to the characteristics described above, the objective of the Process Diagram is to offer a set of concepts, symbols and rules, which if used by the modeller is able to describe all substantial characteristics of real world behaviour in as simple a way as possible.

The Business Process meta-model (see *Chapter 2. Business systems modelling* or [31] for detail) describes the essential concepts of the Process Diagram together with their mutual relationships. At the centre of interest, there are two main concepts:

- stimulus
- activity.

Stimuli are of two main types:

- external (Event)
- internal (State).

Activities are also of two main types:

- *Processing Activity*. The purpose of this activity is to process inputs in order to obtain outputs,

⁸ As is also discussed in the *Chapter 2.1 Conceptual Modelling under the object paradigm*, the common understanding of the term “*conceptual*” tends towards the synonym for “static”. Thus, the “*conceptual model*” concept is usually understood as a “*model of basic terms*” which is naturally static. In the Conceptual Business Processes Model we use the term “*conceptual*” in a slightly different meaning of this word – as a “*model of the substantial elements of the business process*”, i.e. in the sense of “*the process model of the real world*”. In that sense, our use of the term “*conceptual*” is consistent with the static point of view (because from the static point of view the “*model of the real world*” is the “*model of basic terms*” as well).

- *Control Activity* (Decision or Logical Connector). The purpose of this activity is to ensure proper control over the process – a succession of the correct activities according to the internal process state(s) and/or external stimuli and information.
- *Logical Connector* is a special kind of Control Activity defined for the simplification of the model. It is the simplest (primitive) decision, which does not need any information at the input (conjunction and disjunction).

Processing Activity can be either Primitive (i.e. non-decomposable) or Complex. Complex Activity can be broken down into a sub-process (i.e. a set of activities in the form of separate process models), unlike Control Activity, which is in principle non-decomposable.

A description of the process **expresses the way by which the inputs are transformed into the outputs by activities in their defined succession**. Input/Output Sets are of three types:

- *information Set*,
- *Material Set*,
- *Mixed Set*.

The main purpose of such an approach is to *distinguish the **object of the processing** (“material”) from the **information for the processing control** (“information”)*⁹. Therefore, the term “material” is defined in a very abstract manner here. In some specific cases (when describing the business processes of a consulting company for example) the real substance of what is called “material” here can be the information (because the “raw material”, as well as the product of such a company, is the information). Even in such a situation, the need to distinguish between the subject of the processing and the control of the process remains very important.

In addition, the technique supposes the need for modelling a number of external aspects connected to any element of the process, like:

- *Actors* (attendees or “victims” of the process activities),
- *Organisation units*,
- *Problems* related to the process,
- and *any other possible external aspect*.

External aspect may be any particular aspect which is important for any particular reason. As there are so many particular reasons, so many purposes and modelling situations, the *External aspects* concept references a generally unlimited set of potential particular aspects.

⁹ This part of the technique is namely influenced by the ISAC Method (see [15]), particularly by its idea of distinguishing between the data as a subject of processing and the data as control information for processing. We regard the monitoring of this difference as a critical success condition for the successive development of the Information System using the business process-oriented analysis.





Process Diagram and standard process modelling languages

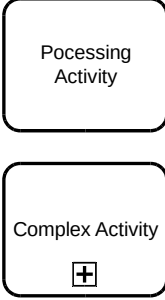


During the last two decades many different standards for business process modelling have been created from different purposes and for different reasons. Most of them are no longer in use, some of them, despite their exceptional quality, have not achieved the position of the widespread standard iDEF ([16]). Nevertheless, two of them can be regarded as dominant:

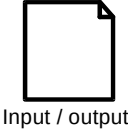
- Business Process Modelling Notation (**BPMN**) ([24]).
- Architecture of integrated Systems Notation (**ARIS**) ([40]).

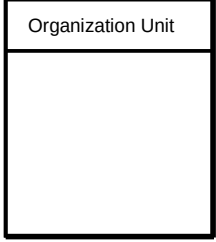
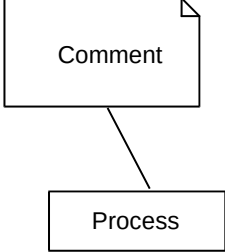
Business Process Modelling Notation (BPMN) as a language for modelling business processes ([24]) is a most important standard fulfilling the above stated requirements for the standardisation. Among other popular standards ([20], [6], [41]) only the BPMN became widely accepted by users as well as by CASE tools producers, and has continued developing in concern to other related significant technology standards OASIS ([22]), Service Science ([39]), and UML ([23]), which is the basic condition for fulfilling the full meaning of standardisation. This fact qualifies the BPMN as a leading professional standard in the field of business process modelling.

Selected basic **BPMN** constructs according to MMABP:

Construct	BPMN Element	Description
Event	 <<Event General>>  <<Event Timer>>	External stimulus for the activity. Information about the event outside of the process and independent of it. <i>it is possible to also use a lot of specific types of the event from BPMN language. Recommended symbols to use are:</i> <ol style="list-style-type: none"> regular business event (Event General), time event (Timer).
Process State	 <<Parallel AND>> Process State  Process End	internal stimulus for the activity. Result of the preceding activity. <i>As BPMN does not recognise the concept of the internal process state it is necessary to use the synchronisation symbol “Parallel(AND)”, which has the same technical nature.</i>

Construct	BPMN Element	Description
		End of the process (pseudo-state).
Activity		Basic element of the process – input(s) to output(s) processing. Activity is decomposable on principle, i.e. it can be always regarded as the process (on the deeper level of detail).
Decision		<p>Elementary (i.e. indecomposable) activity. Decision on the particular follow-up of the process.</p> <p><i>In BPMN, the general decision can be expressed as so-called “complex gateway”.</i></p>
Logical Connector		<p>Either a <i>primitive decision</i> without any information at the input (pre-defined decision) or a <i>logical operator</i> in the expression of the relationship between the various elements of the model (activities or events).</p> <p><i>In BPMN, the logical connectors can be expressed as a “gateway”. There are three basic types of the logical connector in BPMN gateways:</i></p> <ul style="list-style-type: none"> - <i>Exclusive OR</i> represents the Boolean exclusive disjunction (excluding the conjunction). - <i>inclusive OR</i> represents a general logical disjunction (including also the conjunction). - <i>Parallel AND</i> represents a logical conjunction. <p><i>As BPMN does not recognize the essential differences between different modes of use of the gateway, BPMN gateway can have different meanings depending on the modes of its use. Only “1:M split” OR and “M:1 join” AND, used between the process activities are decisions.</i></p>

Construct	BPMN Element	Description
		<i>The remaining possible ways of use of gateways have other meanings.</i>
information Set		input into / output of the process. We recognise three types of the iO set according to their meaning:
Material Set		<i>information Set</i> is a set of the information for process control. Examples: manufacturing plan, strategic investment intention, delivery note, etc.
Mixed Set		<p><i>Material Set</i> is a set of the subjects of processing, i.e. raw material (at the input) or product (at the output), no matter whether it is material or data.</p> <p>Examples: engine component, car (final product) in the case of car manufacturing. Stock list, investment advice (final product) in the case of broker's business (information plays the role of the material here).</p> <p><i>Mixed Set</i> is a set of the combination of the subjects of processing as well as the information for controlling it. Example: delivery together with the delivery note.</p> <p><i>BPMN offers the only way to express the input/output as a paper (information). Various implementations of BPMN in the CASE tools usually offer other possible expressions of the input/output like an “artifact”, “object”, etc.</i></p>
Actor		Abstract person – all kinds of the attendee of the process (person, organisation unit, system, position, profession, role, entity, etc.).

Construct	BPMN Element	Description
Organisation Unit		<p>Unit of the organisation where the process runs.</p> <p><i>BPMN uses the “swim lanes” style for expressing organisational units. Unfortunately it reduces possibilities of organisation structure independent description of the process (what is the basic principle of process management, by the way).</i></p>
Problem		<p>Problem connected to the process in the particular point.</p> <p><i>In BPMN it is generally possible to use the “Note” symbol. Another general possibility is to describe the problem as an attribute of the process or its particular part.</i></p>

BPMN notation offers a rich palette of symbols usable for expressing a number of different forms of events and process states. Figure 12 shows a wide palette of types of an event in BPMN. Besides the main and recommended differentiation of business and time event it is possible to specify the nature of the event as an error, cancellation, rule, link, message, etc. This rich repertoire of event types is the consequence of the “computer oriented” perception of processes, which is very improper in the context of the business system analysis. Therefore we do not recommend use of such classification.

Besides the general event, BPMN specifies the “Start Event”. Respecting the definition of the “event” concept this specialisation is irrelevant. Nevertheless it can be useful for the business analysis of processes where we need to find the original – the only starting event for each substantial business process.

Like in the case of events, BPMN recognises similar types of process end that it calls also “events” despite their opposite nature. The “physical substance” of the process end is rather a state since it is a product of the process, not a phenomenon, which the process is required to react on.

Events	Start			Intermediate			End
	Top-Level	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing
None: Untyped events, indicate start point, state changes or final states.							
Message: Receiving and sending messages.							
Timer: Cyclic timer events, points in time, time spans or timeouts.							
Escalation: Escalating to an higher level of responsibility.							
Conditional: Reacting to changed business conditions or integrating business rules.							
Link: Off-page connectors. Two corresponding link events equal a sequence flow.							
Error: Catching or throwing named errors.							
Cancel: Reacting to cancelled transactions or triggering cancellation.							
Compensation: Handling or triggering compensation.							
Signal: Signalling across different processes. A signal thrown can be caught multiple times.							
Multiple: Catching one out of a set of events. Throwing all events defined							
Parallel Multiple: Catching all out of a set of parallel events.							
Terminate: Triggering the immediate termination of a process.							


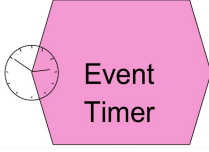


Figure 12: Types of an event in BPMN



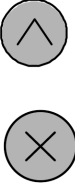
Source: [24], <http://bpmb.de/poster>

ARIS ([41]) is a name for the business process modelling tool and specific notation as well as some connected methodology. Before BPMN became the widely used standard ARIS had dominated this field for many years. Regarding the methodology value and the higher evolution level of ARIS, it has to be regarded as a serious competitor and a great challenger of BPMN: a co-leading language in the field.

Basic methodology requirements for the Process Diagram follow from the idea of basic rules for modelling business processes, which are formally expressed in the Business Process Meta model (see *Chapter 2. Business systems modelling*). The Meta model describes basic elements and their basic relationships and classifications. The following two tables contain the description and explanation of the basic business process model construct and their realisation in BPMN and ARIS notations. From the BPMN notation we use just the basic set of symbols; other complementing symbols are discussed below the first table. Similarly, only a subset of the ARIS notation is used in the second table, and other language possibilities are discussed in the comment column of the table.

Selected basic **ARIS** constructs according to MMABP:

Construct	BPMN Element	Description
Event	 	<p>External stimulus for the activity. Information about the event outside of the process and independent of it.</p> <p><i>ARIS language offers just one type of event. Nevertheless, it is possible to create new kinds of symbols together with their graphical symbols and specific rules for their use (see the proposed Event timer for instance).</i></p>
Process State	 Process State  End State	<p>internal stimulus for the activity. Result of the preceding activity.</p> <p><i>ARIS language does not distinguish between state and event, neither between internal and end states. Nevertheless, it is possible to create new kinds of symbols together with their graphical symbols and specific rules for their use (see the proposed symbols for states).</i></p> <p>End of the process.</p>

Construct	BPMN Element	Description
Activity		Basic element of the process – input(s) to output(s) processing. Activity is decomposable on principle, i.e. it can be always regarded as the process (on the deeper level of detail).
Decision		Elementary (i.e. indecomposable) activity. Decision on the particular follow-up of the process. <i>ARIS language does not distinguish between processing activity and complex decision. Just the primitive decision is regarded as something different from the general activity (see below).</i>
Logical Connector		Primitive decision without any information at the input (pre-defined decision). <i>ARIS language contains just two basic logical connectors for the conjunction and exclusive disjunction logical operations, which are sufficient for an algorithmic description.</i>
Information / Material / Mixed Sets	Many various symbols	input into / output of the process. <i>ARIS language offers a <u>huge palette of disposable types of inputs/outputs</u> including specific graphical symbols. Moreover, this palette is not limited in principle; it is possible to create own objects together with their graphical symbols and syntactic limitations (i.e. allowed associations of allowed types to allowed types of other objects).</i>
External aspects	Many various symbols	Any aspect which can be associated with any element of the process description in principle. <i>ARIS language offers a <u>huge palette of disposable external aspects</u> including specific graphical symbols. Moreover, this palette is not limited in principle; it is possible to create own objects together with their graphical symbols and syntactic limitations (i.e. allowed associations of allowed types to allowed types of other objects).</i>

As it is not critically important which particular language is used for the detailed process modelling if it fulfils the basic methodology requirements (as they are expressed in the meta-model); all of the following examples in this book are in the BPMN notation.

Main features of BPMN and ARIS languages

During the process of the business process modelling, no matter which particular standard language is used, it is necessary to respect basic methodology rules following from the nature of a business process as it is described in the first two chapters. These rules are also expressed above in the form of the Process Diagram Technique. These rules are expressed most generally as well as precisely in the business process meta-model (see *Chapter 2. Business systems modelling*). Nevertheless, it is important to know specific features of the language in use in order to exploit it well and correctly. Especially important is the notion of limitations and insufficiencies of the given language as it can lead to methodologically incorrect models.

The main features of both standard languages are:

BPMN is a relatively new language which has been developed to be a standard. As a standard it is widely supported with modelling tools, and it can be supposed this support will continue in the future.

BPMN is strongly oriented on the technical side and formal aspects of business processes and ignores important business aspects of them (expressed in the BP Meta-Model). These are its general weaknesses. Moreover, in spite of its technical orientation it is not exact enough to be usable for formal specification of the purpose of the use of technology in processes (including the idea of “automation of processes”). The definitions of its concepts are not mutually consistent, therefore it cannot be based on the formal Meta-Model. Some important BP modelling concepts are not supported in this language. On the other hand, there are many concepts irrelevant to the general purpose of BP modelling in terms of the idea of the process-driven management.

ARIS is a traditional and matured language which is based on the precise and relatively matured methodology. As a former de-facto standard for BP modelling it is supported by many modelling tools although it can be supposed this support will decrease in the future in comparison with the support for BPMN.

ARIS supports the business-oriented as well as the technical view of business process. It offers expression of many important business aspects in many mutual relationships with the support of methodology ensuring their logical consistency. Nevertheless, not all important business aspects including the idea of process-driven management are sufficiently supported in this language. Similarly like BPMN, even ARIS prefers technical aspects of business processes regarding them only as a way to information system development (in accordance with its name). The dark side of its strengths and competences in description of business aspects is its relatively high complexity. It may cause serious problems with the use of its functionality, which may disqualify ARIS in the battle for being the widely accepted standard.

The most important **common insufficiencies** of both languages from the MMABP Meta-Model point of view are:

- **insufficient support of the concept of process state** in the process description. BPMN does not understand this concept at all, while ARIS respects this concept but does not respect its natural difference from the concept of “event”.
- **insufficient support of the concept of event** in the process description. BPMN obscures the business substance of “event” by offering too many types of it, most of which are irrelevant to its business substance. On the other hand ARIS does not offer more than the general concept “event” even for the concept of “state”, which obscures the business substance of the “event” as well.
- **insufficient conception of the global model of processes (process map)**. BPMN does not understand the importance of this model at all. ARIS does not understand the real meaning of the process map, presuming sequences of processes which contradict with the idea of process-driven management.

The above mentioned common problems in both languages can be understood as challenges for the future development of standards in the field of modelling business processes. Today it seems that **BPMN is predetermined for being the standard** as it was born as a common (not private) standard and much effort has been paid to its marketing and “political” support of its formal position. On the other hand **ARIS is a much more mature and methodology supported language** with a better position on the way to the ideal language in terms of the ideas of process-driven management. No matter how the future development will go, the above stated challenges for the language development should be fulfilled if the business process model fulfils the idea of process-driven management.

For detailed analysis and comparison of both leading process modelling languages see ([36]).

intentionality and a crucial role of process stimuli and activities

In the detailed process model, MMABP requires a strict respect to so-called *process states*. The need for process states follows from the fact that the business process always represents some intention (see also *Chapter 1.1 Business system as an equilibrium of intentions and causality*).

In the legendary article [37] Norbert Wiener expressed the idea which fatally influenced the later development of cybernetics: “*all purposeful behaviour may be considered to require negative feed-back*”. *The concept of negative feed-back is explained there as follows: “...the behaviour of an object is controlled by the margin of error at which the object stands at a given time with reference to a relatively specific goal. The feed-back is then negative, that is, the signals from the goal are used to restrict outputs which would otherwise go beyond the goal.”*.

According to the basic work in the field of process-driven management ([7]) business process always follows some goal. The goal is a fundamental attribute of a business process as it is regularly used in matured methodologies like in [6] for instance. That means that **business process is always an intentional process**. By the term intentional process we mean the process of **purposeful behaviour of interested object that follows some goal**.

Concluding from previous two paragraphs we can find that the business process, as it is an intentional kind of a process, **has to have some negative feed-back** which ensures restriction of its outputs in order to keep them in the margins of its goal. This characteristics strongly distinguishes the business process from the process in general (i.e. in just technical /physical sense) as well as from processes which do not need any feed-back like machine-managed or automated processes running without a contact with their environment.

In case of business process, **feed-back means some input to the process from its environment which is causally connected with some process output**. The value of the input should influence the following behaviour of the process in terms of keeping it in the margins of its goal. This means that “intermediate” inputs to the process (i.e. none-starting inputs to the process coming between its starting and end points) are critically important parts of the business process distinguishing it from other, non-intentional (i.e. non-business), processes. Working with processes we have to take into the account even the time dimension; every input to the process from its environment has to be synchronized with the process run. Thus in each part of the process where some input which will influence the following process run is expected the **process state** has to be placed. As it is explained earlier in this chapter, process state means such a point in the process structure, where nothing can be done before the input to the process occurs, i.e. point of waiting for the input. The concept of process state is present just in some process modelling standards (like iDEF, see [20]), partially present in some others (like ARIS, see [40]), many standards do not support it. Widely accepted process modelling standard BPMN ([24]) does not recognize this concept at all.

Regarding the importance of the above outlined problem together with the insufficient support in most of process modelling standards it can be said that the primary task for every process modelling methodology is to **allow the modelling of process states ensuring the critically important presence of the negative feed-back** no matter which notation and/or modelling standard is used.

Basic Process Flow Pattern

The Basic Process Flow Pattern expresses the basic structure of the process model which respects the essential rules of the MMABP methodology. These rules express the "technical" necessities which mainly follow from the general theory of algorithms as well as the specific aspects of the business process which distinguish the business process from a process in general (i.e. just technical) sense. The lately mentioned rules follow from the theory of BP management and re-engineering which is anchored already in the basic work in this field: [6].

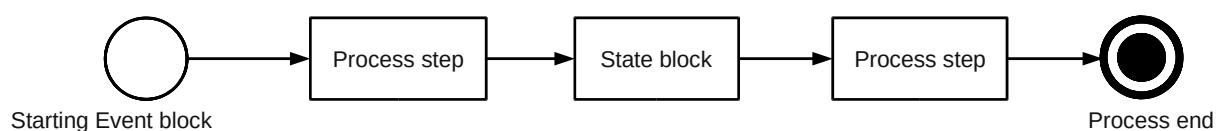


Figure 13: MMABP Basic Business Process Flow Pattern

Source: Author

Basic Process Flow Pattern (see Figure 13) expresses the essence of the process flow using three methodically essential types of the process elements: events, steps, and states.

According to the Basic Process Flow Pattern the business process should be described as a sequence of *Process steps* interrupted by *State blocks* starting with just one *Starting Event block* and resulting in one or more *End states*. Figure 14 illustrates an exact definition of these basic blocks. The definition is written in the semi-formal metalanguage based on the simplification of the standard Extended Backus-Naur Form. Used meta-symbols have following meanings:

- $A = [\text{element1} \mid \text{element2} \mid \text{element3}]$ means that the item A can be either *element1* or *element2* or *element3* exclusively.
- $A = \{ \text{elementX} \}$ means that the item A consists of one or more *elementsX*.

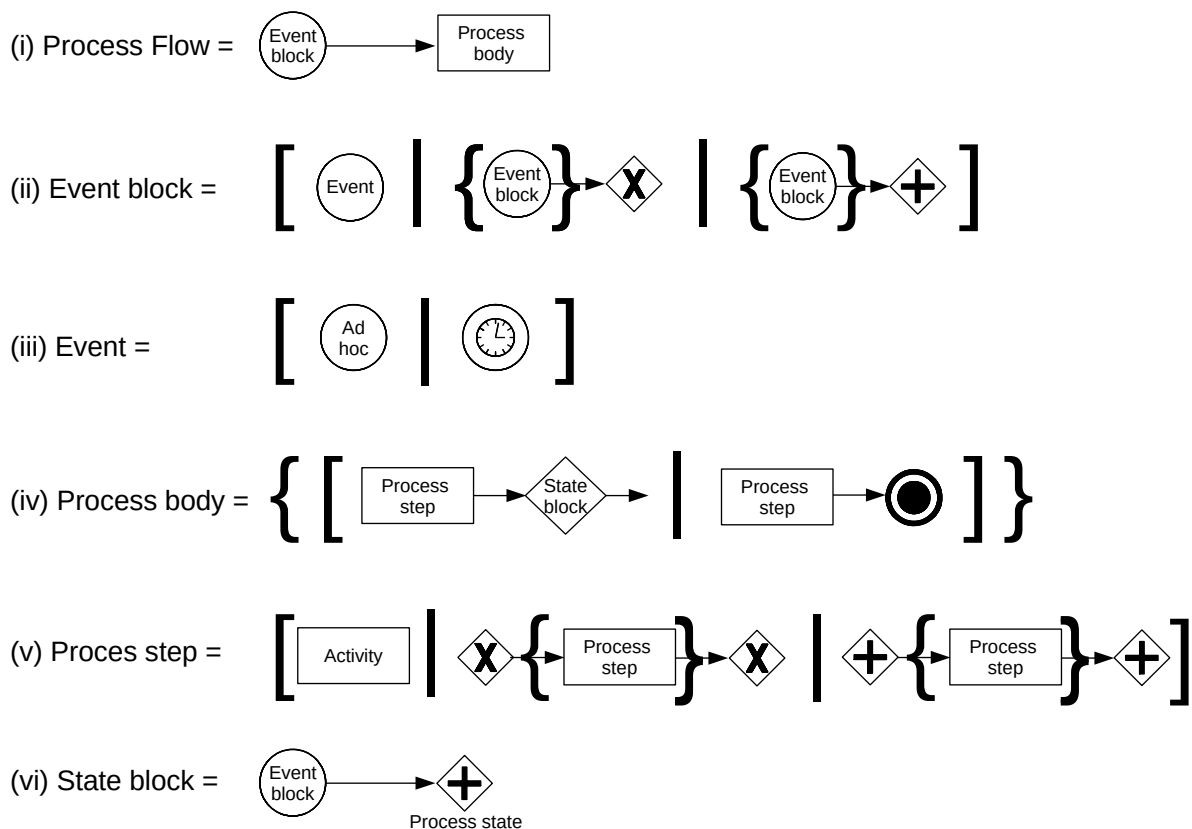


Figure 14: Definition of basic blocks and concepts of the Business Process Flow Pattern

Source: Author

Particular definition sentences can be read as follows::

Def (i): Process flow begins with starting Event block followed by the Process Body.

Def (ii): Event block is either a single event, or structure of mutually exclusive Event blocks, or structure of mutually synchronized Event blocks.

Def (iii): Event can be either an ad-hoc event or a timer.

Def (iv): Process body consists of one or more pairs where each pair consists of a Process step followed by either State block or End State. If the pair ends with State block the description should continue with another pair (see the arrow after the State block). End state always means the end of the process.

Def (v): Process step is either a single Activity, or structure of mutually exclusive Process steps, or structure of parallel Process steps.

Def (vi): State block is a synchronization of internal process flow with expected event(s) expressed as an Event block (in other words: waiting for the event(s)).

Event block represents the external influence which the process always has to respect. It works either as a trigger or a limiter of the process. In both cases it has to be unambiguous which means, among others, that it has to represent a single point of time. Therefore it can be either a single event or a time-elementary structure of events. If it is a structure it can express either the synchronization of parallel events (event blocks) or the set of possible mutually exclusive alternative events (event blocks) in order to be time-elementary.

Process step represents an action element of the process on the *level of process steps*¹⁰. It can be either a single activity (step) or a structure of process steps (and consequently a structure of activities). Similarly as in the case of event also an activity should be unambiguous. Therefore, if it is a structure, it can express either the synchronization of parallel process steps or the set of possible mutually exclusive process steps. It cannot express a sequence of process steps as it would be a violation of the elementariness rule. The methodical reasons and meaning of the need for the elementariness of activities in the process description is discussed in more detail below.

¹⁰ MMABP distinguishes two levels of a detailed description of the process: *process step* and *process activity* levels. For details, see Chapter 3.3 Process memory and MMABP process abstraction levels.

State block represents the essential need to synchronize the process run with expected events. This need follows from the fact that the event is always an objective external influence and thus it must be respected. From the physical point of view such respect means synchronization – waiting for the event (event block). As the BPMN notation do not recognize the concept of process state there is no other way than to express the process state with the general symbol for synchronization – the "AND gate". In order to distinguish between the general synchronization and its specific meaning as a process state we complete the BPMN with the stereotype <<*process state*>>.

One of the most important ideas expressed in this pattern is that *there can not be a sequence of process steps uninterrupted by the process state*. This rule reflects the essence of the **definition of an elementary process step**:

(a) the process activity is regarded as an elementary **process step** if there is no objective reason for its interruption,

(b) the reason for the interruption of the activity is objective if it comes from outside of the process.

Rule (b) of this definition means that each objective reason for the process interruption is actually represented by an event (external influence). Thus, any set of activities of the process, no matter how technically complex it is, must be regarded as an elementary step if there does not exist an external influence (event), which the process has to respect (i.e. wait for). This consequence well illustrates the fact that the elementariness of a business process step is not only its physical but much more a functional attribute as the business process itself is always more than a physical process (algorithm) only. This way, the methodology prevents the analyser from the pointless unlimited dividing of the process activities which is a frequent mistake in the field of BP modelling. The necessity of such safety fuse in the methodology against the unlimited division of activities is given by the fact that aggregation is a dominating type of abstraction in the field of process-oriented modelling (unlike in the field of object-oriented modelling, where the generalization is a dominating type of abstraction). This fact manifests itself in the principally unlimited possibilities of division of activities known as a rule: *any single process activity can be decomposed into the structure of sub-activities – a process* (as it is also defined by the process meta-model at Figure 9). As the division of activities is physically unlimited, the methodology has to define some logical – functional definition of the very low level: the level of the process step elementariness.

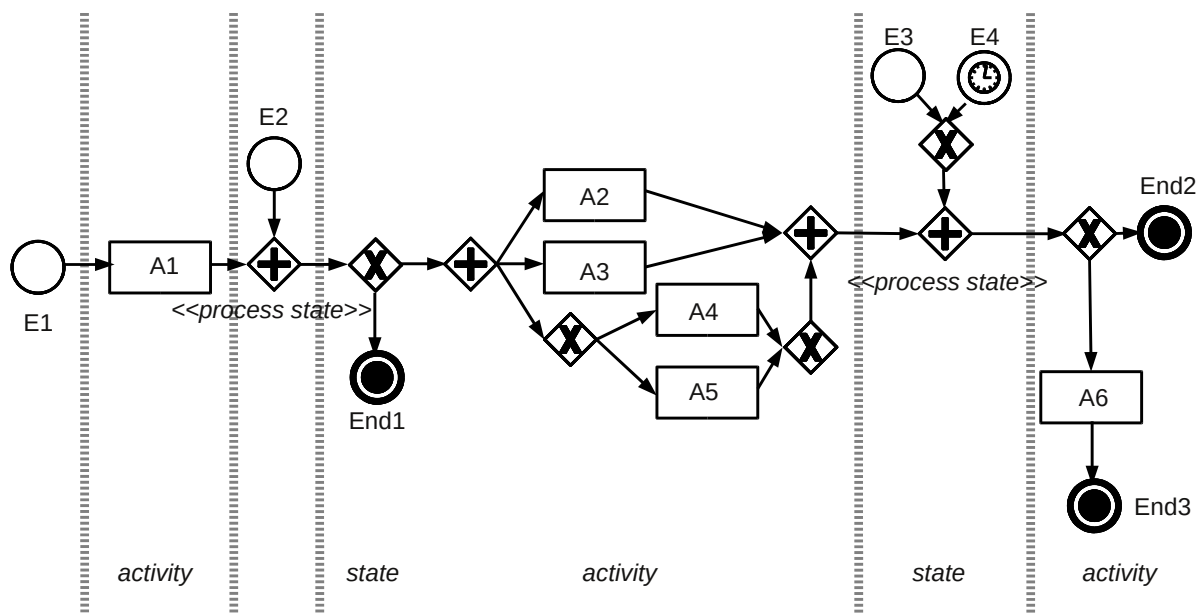


Figure 15: Correct business process flow example

Source: Author

Figure 15 shows a symbolic example of the process which can be regarded as correct according to the Basic Business Process Flow Pattern. The process can be seen as a sequence of several parts each representing one block of a particular basic type (see the division of the whole process by vertical dashed lines). It is beginning by the starting event block which consists of just single event E1 in this case – the starting event of the process. The starting event is followed by the process step consisting of just single processing activity A1. According to the pattern, the process step is followed by the state block in the form of synchronization of the process run with just a single event E2. Following process step represents more complex structure of activities: it consists of two main alternatives: either the process end End1 or the structure of three parallel activities where the first two are single processing activities A2 and A3, and the last one is a structure of two alternative processing activities A4 or A5. Following state block represents the waiting for two alternative events E3 or E4. The last process step is a structure of two alternatives: the processing activity A6 followed by the process end End3 or the immediate process end End2.

The example at Figure 15 illustrates that and how any algorithmic structure of the process can be checked whether or not it fulfils the basic definition of the business process expressed by the Basic BP Flow Pattern: *the business process is a sequence of Process steps interrupted by State blocks starting with just one Starting Event block and resulting in one or more End states.*

Events, states and activities of the process play a crucial role not only in the process model. One of the most significant consequences is that the states serve as a “meeting point” of the two main points of view existing in real world modelling:

- object model (static – structural model of the real world),
- process model (dynamic – behavioural model of the real world).

Therefore, we regard stimuli and activities as very important aspects of the process. They enable both interconnection between the object and process models, and the expression of the appropriate integrity rules.

In the process model, *states of the process* (processes) represents a particular point within the process – the place between two particular activities. From the point of view of the first activity, the state is a *result of the activity*. From the point of view of the second one, the state is a *stimulus for the activity*.

In the object model, *states of the objects* are described. The state represents a particular point of the object life cycle – a place between two particular actions of the object. From the point of view of the first object action, the state is a *result of the action*. From the point of view of the second one, the state is a *starting point for it*.

It is obvious that the states of the process should somehow match the states of the relevant objects (i.e. those objects which are related to the process). In addition, the activities of the process, which cause some effect outside the process (i.e. Processing Activities), should also match the actions of the relevant objects. And, lastly, there is no doubt that the real world events that work as stimuli for the process activities should also, somehow, affect the relevant objects (as triggers of the object’s actions).

The following two tables outline the basic requirements for the consistency rules following from the existence of these two main points of view.

The first table focuses on the external facts that have a different meaning in each of the viewpoints. The second table focuses on concepts existing in both viewpoints and having a specific meaning in each of them.

Table 2. Outline of the consistency rules requirements concerning external facts (different meanings of the same fact)

Fact	Object Model	Business Process Model
Event	Stimulus for: <ul style="list-style-type: none"> object internal state change, possible communication with other objects (send the message) in the case of the “common action”. 	Stimulus for: <ul style="list-style-type: none"> operation execution, process state change, output production, possible communication with other processes (process co-ordination).
Output	Consequence of <ul style="list-style-type: none"> object action, object internal state change. 	Consequence of: <ul style="list-style-type: none"> operation execution (product), process state change.

Table 3. Outline of the consistency rules requirements concerning internal concepts (different meanings of the same concept)

Concept	Object Model	Business Process Model
Action	Action executed/allowed by the object Causes: <ul style="list-style-type: none"> object state change, possible output production, possible communication with other objects (send the message) in the case of the “common action”. 	Activity inside the process Causes: <ul style="list-style-type: none"> process state change, possible output production, possible communication with other processes (co-ordination of processes).
State	Object life cycle state <ul style="list-style-type: none"> starting point for action processing, result of action processing. 	Process course state <ul style="list-style-type: none"> starting point for operation execution, result of operation execution.

In the following text, the facts described in both tables are used for the formulation of basic consistency rules, addressing the relationships between the object and process models (see *Chapter 5 Consistency of business system models*).

3.3 Process memory and MMABP process abstraction levels

In controlling complex processes (which often have complex relationships to other processes) there is a need to store the information about the actual process state. It is a vital and conceptual condition of each process control – in the computer model of the process (i.e. conceptual need of Data Stores) as well as in the real world process implementation (the need for traditional paper evidence, for example).

Such a need occurs even in object-oriented analysis and design methodologies. For example, Michael Jackson [10] offers an excellent understanding of this fact. In some object-oriented methodologies, this principle is called “the object memory” ([38], [5]).

Using the analogy in the OO modelling methodology, we call this principle “the process memory”. In the concept of “process memory”, we not only include the attributes of the actual state of the process, but also the data gathered by the activities. Once the data are gathered, they exist inside the process and can be used by its activities without limitation (global data access). This rule also significantly reduces the complexity of the process description.

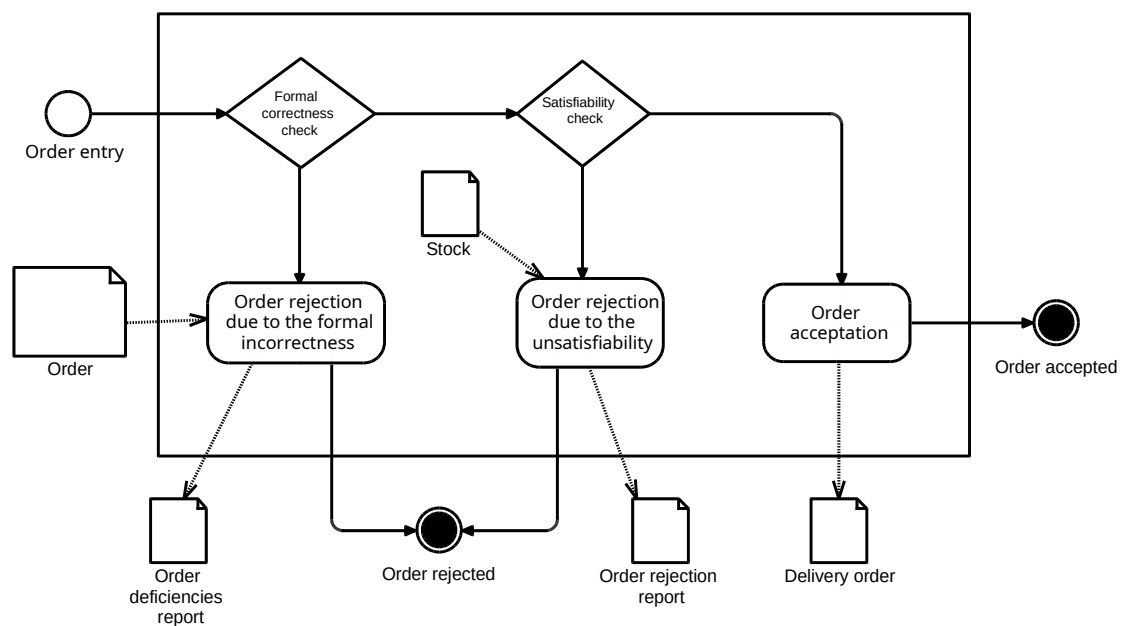


Figure 16: Example of a primitive process (Order Receiving)

Source: Author

The need to store the information about the process current state also serves as the criterion for distinguishing between primitive and complex processes. When there is no need

to store the information about the state of the process, the process is so simple that it is possible to take it (and also implement it) as a single algorithm. The need to store the information about the state of the process always indicates the possible parallelism inside the process, or at least in the communication with other processes.

See Figure 16 for an example of a primitive process.

For an example of a complex process see Figure 17. In this example, the simple process “*Order Receiving*” from Figure 16 occurs as a single activity. It is obvious that there is a need to store the information about the state of the process between each two succeeding activities. The state “*Order accepted*” describes the need to wait for the goods’ dispatching, and the state “*Goods Delivered*”, describes the need to wait for customer payment. Both situations indicate possible communication with other processes or actors.

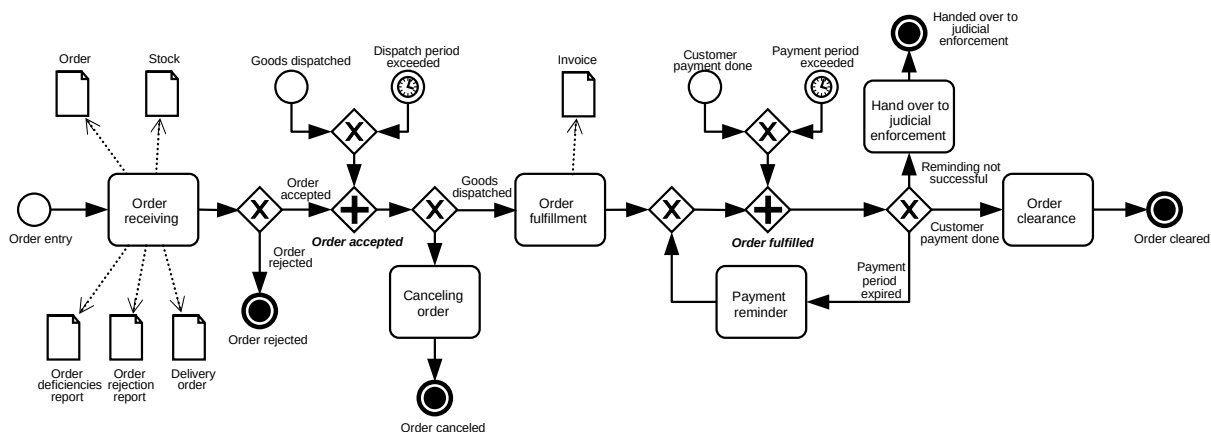


Figure 17: Example of a complex process

Source: Author

The concept of *process state* discussed in the previous chapter together with the concept of process memory, are essentially related to the aspect of collaboration of processes and this relation directly determines the needed level of detail of the process algorithmic description – so-called the level of *process steps*. Nevertheless, collaboration of processes is not the only crucial aspect that needs to be taken into account and consequently, process step level is not the only needed level of detail in the process algorithmic description. Moreover, respect for the different aspects in the process modelling causes the need to distinguish also different levels of detail also in the system process model as it is lightly discussed also in *Chapter 3.1 Modelling the system of business processes (Process Map)*. In this chapter, we explain the meaning and reasons for MMABP process abstraction levels.

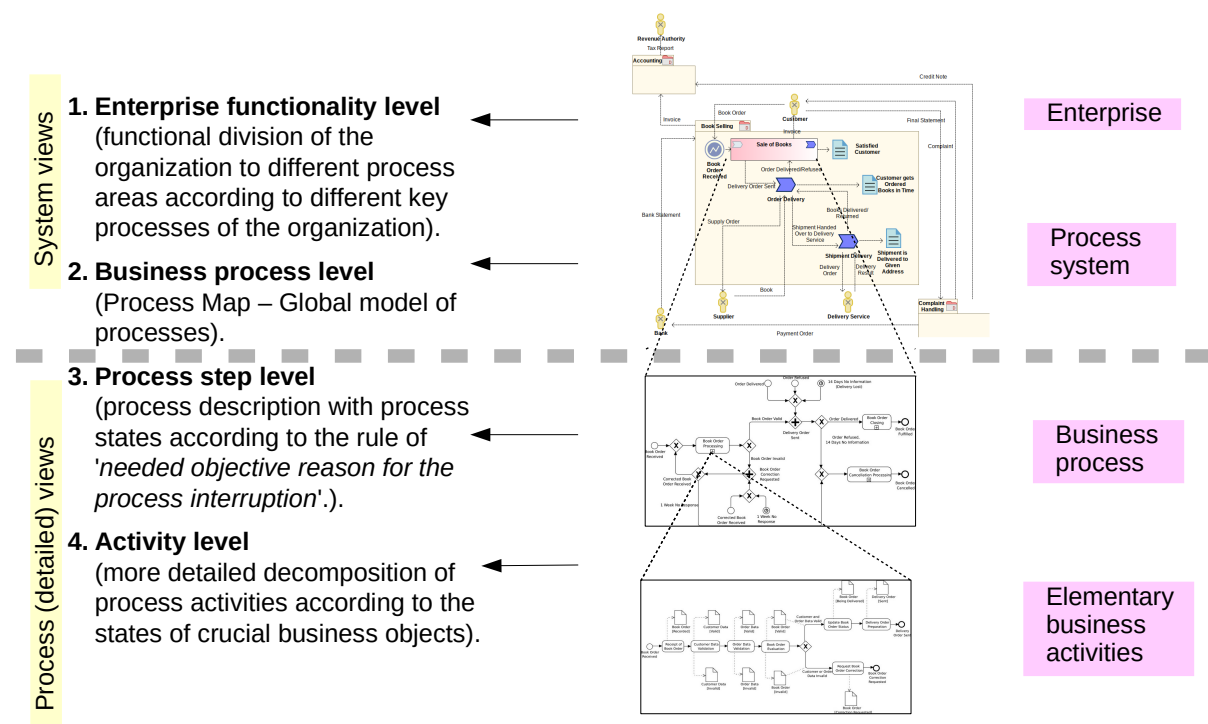


Figure 18: MMABP process abstraction Levels

Source MMABP A4 Method

Figure 18 shows all four MMABP process abstraction levels and their structural relationships. There are four levels of abstraction, two for the system (global) view of processes and two for the detailed view of the process flow.

The two **system (global) models** are:

- **Enterprise functionality model** that describes the functional structure of the business. This way of structuration of the business activities is close to the traditional view of management and allows the analyser to apply the principle of process-oriented structuring of the business on the proper/optimal parts of the complex business system¹¹ - so called *business domains*.

¹¹ The traditional management naturally leads to the grow of enterprises over the optimal size mixing in one enterprise many various and very different business fields. On the other hand, process-driven management naturally leads to the division of the business system to the optimal – homogeneous parts, where there is usually one key process supported with several support processes. Such a division is also called “downsizing” as it typically reduces the extent of the business unit. Therefore, applying the process-driven structuring to the whole traditional enterprise does not make sense from the process-driven management perspective.

- **Process Map** that describes the system of processes characterising the behaviour of actors in one business domain. The processes in the Process Map are of two essential types: key and support processes. That means that there are local key and support processes in every business domain, no matter whether this domain plays a key or supporting role in the whole business system. This relativism supports the process-oriented thinking like the *Principle of outsourcing* for instance¹².

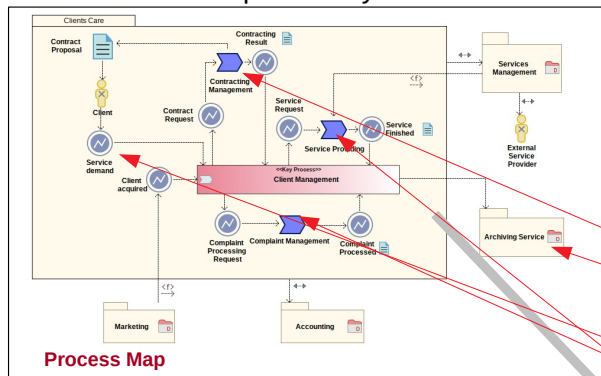
The two **detailed models** are:

- **Process steps model** that describes an algorithmic logic of the process that is driven by the collaboration with other processes and actors. The process step level can be expressed with a simple rule: “*the only reason to divide the process step into more steps can be a possible external influence*”. From the process perspective, the external influence means an event (set of events), which the process has to be waiting for. Such a place in the process represents a *process state*. For a detailed explanation of the concept of process state see the part *Basic Process Flow Pattern* in *Chapter 3.2 Modelling business process details*. The need to keep the basic description of the process flow on the level of process steps follows from the need to directly respect the Process Map, expressing the needed communication with other processes as the Figure 19 illustrates.
- **Process activity level model** that describes an internal algorithmic structure of one process step. The need to describe the internal structure of the process step follows from the need to directly respect the causality of the business system in the process flow. Unlike the process steps model, which reflects the nearest higher level model (Process Map), the activity level model reflects the model of the business system causality, especially so called models of the life cycles of objects. The detailed explanation of the the activity level of process description follows.

The need for the description of the process on the level of activities follows from the need to directly respect the causality of the business system in the process flow. For instance, let us imagine that we decide to replace the service of the *Complaint Management* supporting process (see Figure 19) in the *Client Management* process (see Figure 20) with just a single step *Complaint handling* as Figure 21 shows. Such a change may be explained that there is no need for the communication with other processes nor actors during the processing of complaint.

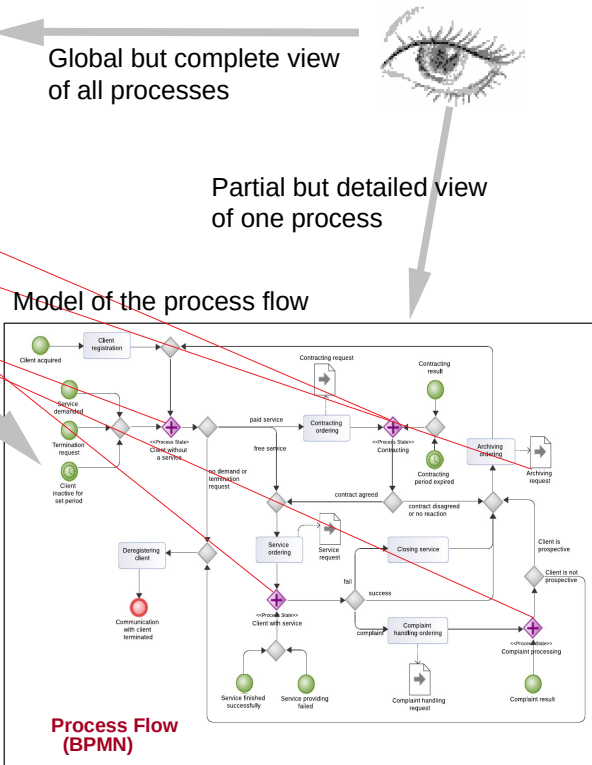
¹² The Principle of outsourcing states that every process element of the business (business domain, business process) represents the business service (set of services), no matter whether it is implemented as a part of the organization or is being bought as an external service. It is because the process element represents the contents, not the implementation, which also follows from the needed flexibility of a process-driven organization.

Global model of the process system



Structure of the system.
- processes with their mutual associations:
communication of key and supporting processes

Basic logic of the process flow
- algorithmic logic of the flow of the process



Global but complete view of all processes

Partial but detailed view of one process

Model of the process flow

Process Flow (BPMN)

Figure 19: Correspondence of Process Map and Process Steps models
Source: Author

The contents of the *Complaint Handling* sub-process shows the model at Figure 22. Complaint is either accepted and then the company should make some compensation or complaint is not accepted. In the case of non-accepted complaint there is a standard suspicion of a dishonest acting of the customer therefore, the customer must be “re-evaluated”. The result of the re-evaluation may be either an exoneration of the customer or the classifying the customer as “not prospective” that consequently leads to his/her de-registration (see Figure 20).

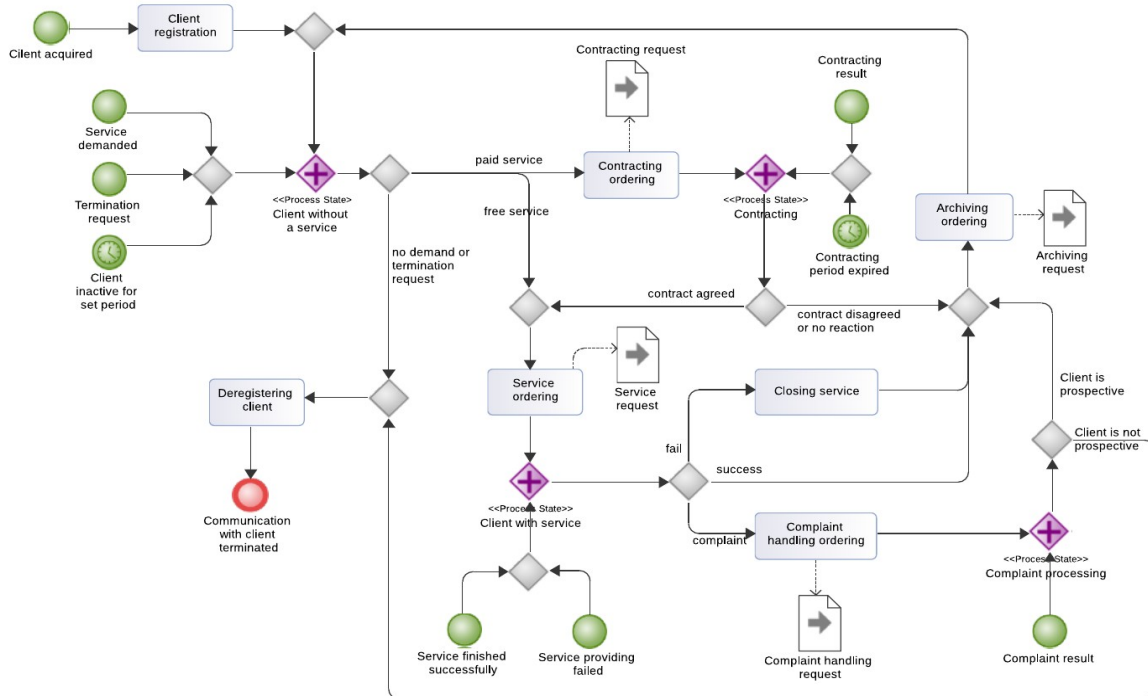


Figure 20: Process steps model of the process Client Management

Source: Author

The decision about the prospectiveness of the client that follows the *Complaint Handling* step in the modified model at Figure 21 then represents some internal decision (or set of decisions) made inside the *Complaint Handling* step as a result of the set of checking actions. To explain the contents of the decision(s), we need to describe this set of actions as a sub-process. Particular decisions in such a sub-process then correspond to the particular states in the life cycles of the corresponding object(s) (including the ends of the life cycle – so-called end states), which are the *Complaint* and the *Customer* objects in this case.

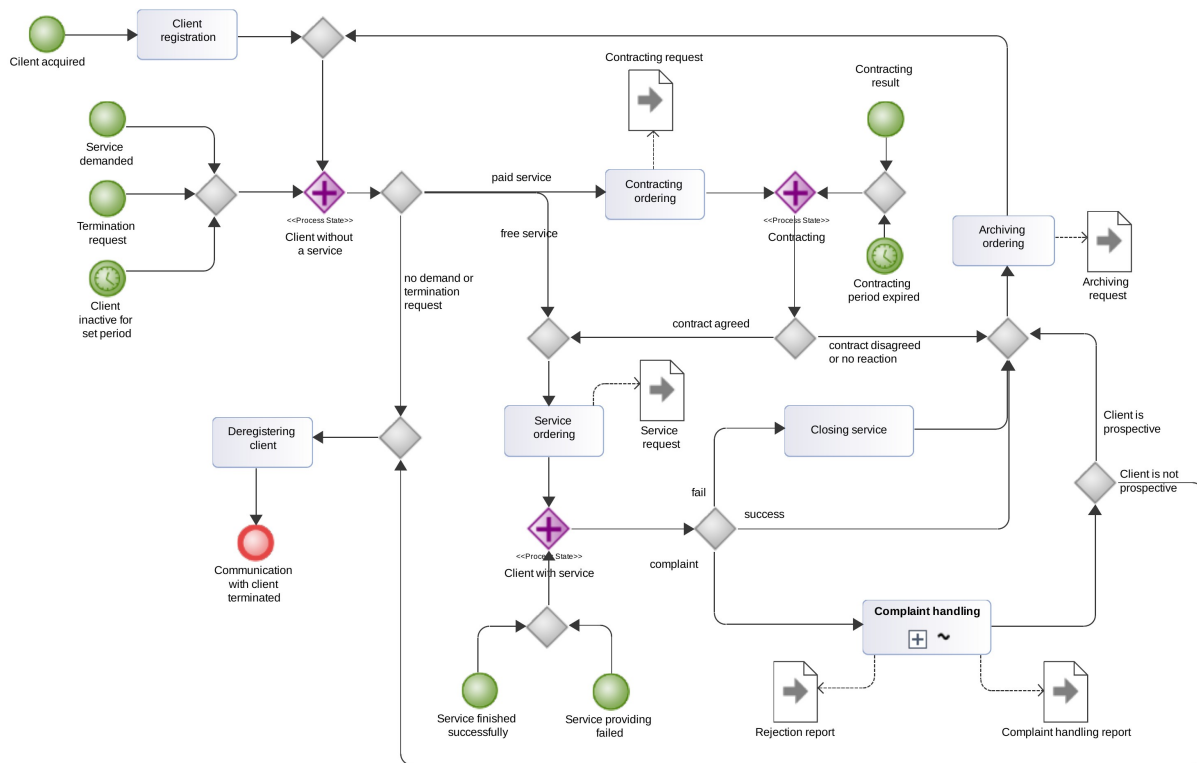


Figure 21: Modified Client Management process

Source: Author

Particular alternatives in these decisions consequently correspond to the particular transitions between states of the *Complaint* and/or *Customer* objects. For instance in this example, the life cycle of *Complaint* contains mutually exclusive states “Accepted” and “Not accepted”, and the life cycle of *Client* contains mutually exclusive states “Prospective” and “Not prospective”. Operations related to the transitions to these states in the life cycles then represent the contents of the actions and decisions in the sub-process *Complaint Handling* (see Figure 22).

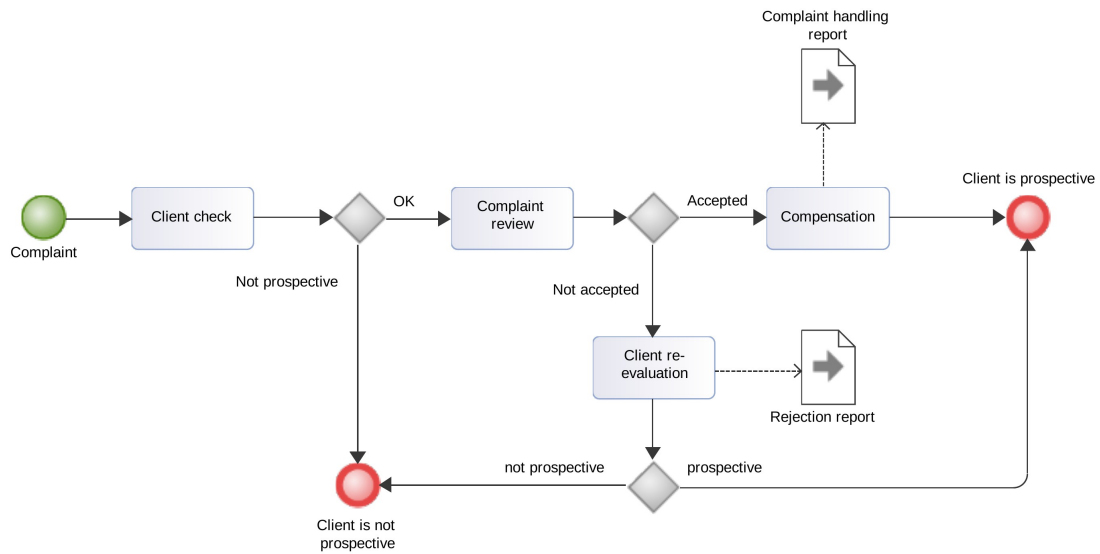


Figure 22: Activity-level sub-process Complaint Handling

Source: Author

In this way, the activity level of the detailed process model reflects an important causality of the business system that determines the decisions in the process, which is actually described in the life cycle models. Therefore, MMABP requires the description of an internal process structure of the process step (an activity-level process) if the decision(s) in the process that follow from this step are based on the causality of the business system, i.e. they are based on the objective facts that all possible processes have to respect.

Chapter 4. Modelling business objects

In this chapter, we explain how to model business objects that characterise the business system to be described. Unlike the model of business processes that describes the behaviour of actors in the business system, the model of business objects determines the extent and basic characteristics of the business system that we usually call the *business system logic*. MMABP uses for that purpose two basic types of models:

- System model of objects (UML Class Diagram) alias the conceptual model that represents the description of the *modal logic of the whole business system*.
- Detailed model of object (UML State Chart) alias the life cycle of the object that describes the piece of *causality of the business system* related to the particular object.

In this way, MMABP can cover not only the modal logic like the standard conceptual modelling methods do, but also the important part of the causality of the business system since both they determine the possibilities to achieve the business goals and consequently, naturally constraint the possible contents of business processes.

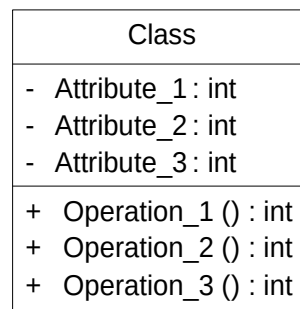
For the broader context see also *Chapter 1.1 Business system as an equilibrium of intentions and causality* and *Chapter 2. Business systems modelling*.

4.1 Modelling the system of business objects

We understand the term “system of business objects” to mean the global view of objects. This model expresses which objects in which mutual relationships form the business system. For an example of a particular system of business objects see Figure 23 below. In principle, the system view of objects can simply recognise their existence and mutual context, not their dynamic details. This model is a conceptual model in the traditional meaning of the term. Particular object classes in the model represent concepts which identify possible real objects of the business system. Relationships among object classes then identify possible links among real objects of the business system. Both real objects and their mutual links are dynamic in the real world; they are naturally changing in time. These dynamic aspects cannot be described in this model as it principally only represents a static view of objects. The detailed objects model is intended for such description (life cycles of objects).

Class Diagram

The standard tool for conceptual modelling, which allows linking with the detailed object models, is the Class Diagram from the UML [23]. Class Diagram describes the static structure of the system in the form of classes and relationships among them. It is a basic diagram of the whole UML diagrams repertoire.

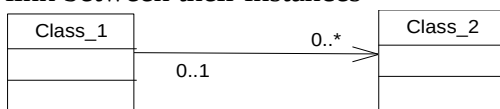


Basic described elements of the Class are:

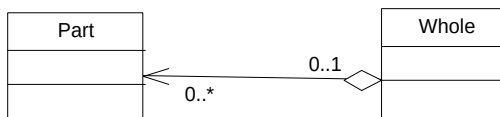
- attributes of the class
- operations – actions/methods of the class

Basic kinds of relationship are:

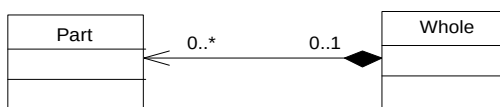
- association – general semantic relationship between model elements, specifying the link between their instances



- aggregation – form of association expressing the relation between a part and a whole



- composition – strong aggregation. Existence of all parts is dependent on the existence of the whole. A particular part can only belong to the one particular whole.



- generalisation (inheritance) – hierarchical relationship where the child class is a specialisation of the parent class (i.e. “inherits” the attributes and operations of the parent class). The child class can also have its own attributes and operations. Inherited operations can have different implementation in different children but their meaning is the same (this fact is called “polymorphism”).



In the object-oriented view generalisation dominates the second basic type of hierarchy: aggregation. In practice, it means that generalisation should be regarded as a principal general meta-quality of an object, while aggregation is just one of possible kinds of relationship between object classes¹³.

UML recognises some additional types of relationships in the class diagram: dependency and realisation, which are intended for specific purposes in the field of modelling computer applications and thus are not relevant for the general use of this diagram in terms of conceptual modelling.

Multiplicity of relationship between two objects can be one of the following types:

Expression	Meaning
0..1	0 or 1
0..*	0 and more
1..1	Just one
1..*	1 and more
*	0 and more

¹³ In fact, generalisation should not be regarded as a relationship (despite of the fact that it is regarded as such in the UML) as it does not represent more than one object, even though it is related to both the generic and the specific concepts. Both concepts nevertheless represent just one object because the specific concept is a (specific) kind of the generic one. Therefore generalisation is usually called “ISA Hierarchy” in the field of conceptual modelling. In the case of generalisation it is critically important to distinguish between the concepts *concept* (alias *class*) and *object*, as one physical object is represented here by two different concepts (the generic and the specific).

A simple example of the system of objects model can be found in Figure 23. It is a fragment of the global object model of the university environment. Of course, as it is just a fragment of the complete global object model there are many, possible important, objects and relationships missing (for instance: *Administration Employee* undoubtedly has many other roles than just an *Administrator of course*; it has more relationships to objects which are not present in this model, or even to objects present there (like the *Project* for instance)).

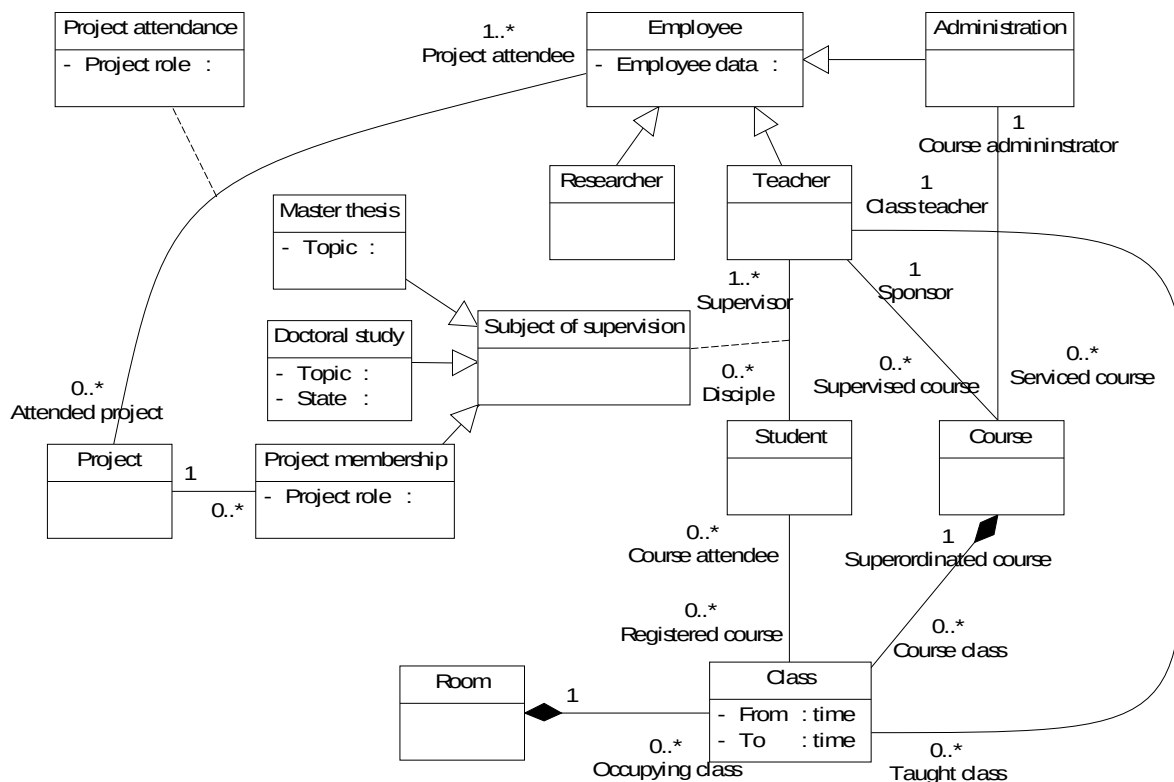


Figure 23: Global model of objects (Class Diagram)

Source: Author

This example also shows an additional important feature of the UML Class Diagram: the so-called association class. Association class is a class used for the expression of details of an association. Strictly viewing it is not a class in fact, but an association¹⁴. UML cannot

¹⁴ The difference between *class* and *association* is essential; *class* always represents a unique identity while identity of *association* is given by the identities of associated objects. Nevertheless, in the conceptual model the model creator decides how to express the Real World facts and in most cases there are several different ways to do it (as a stand-alone object, attribute, or association), which mutually differ in their level of generality and exactness. This is called *semantic relativism* in the conceptual modelling theory.

express attributes of association, neither can it express an association to association. In such situations the association class can be used. There are two association classes in this example: *Subject of supervision* and *Project attendance*. The reason for expressing both relationships as classes is the need to connect some additional information to them. *Subject of supervision* is particularly a generic concept having three specific forms (*Master Thesis*, *Doctoral Study*, and *Project membership*), each of which has its specific attributes and relationships. The *Project attendance* relationship, on the other hand, needs to express the role which the *Employee* plays in the given *Project*¹⁵.

4.2 Modelling business object details

Similarly to the business process models, even in the field of business objects there is a need for a detailed view on some particular objects. Like in the case of business processes, even the detailed view of a business object means viewing the object as a process. This process represents everything that can happen during the lifetime of the particular instance of the object class. Therefore this detailed model of an object is called the **object life cycle**.

The object life cycle expresses the internal dynamics of each object of given class. It describes the mechanism of the object evolution during the time. As the tool for the Object life cycle description, the methodology uses the State Chart diagram from the UML (UML (2010)).

State Chart

As Figure 24 shows, the State Chart is a tool for **describing possible (allowed) states of the object together with the possible transitions among them**.

Each transition is described with two attributes:




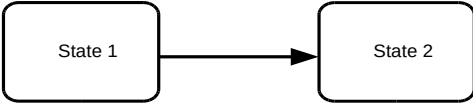
- **reason** for the transition,
- **method** of the transition realisation.

MMABP regards the State Chart as the most suitable tool from the Unified Modelling Language for the purpose of the object life cycle description. Nevertheless, the State Chart has not been originally intended as a tool for description of life cycle. Its roots are in the field of state machines theory, and it is closely connected with the concept of so called “real-time processing”. However, the concept of the state machine in general is not substantially

¹⁵ By the way, the similarity of *Project attendance* and *Project membership* concepts shows that the *Project role* attribute tends to be a stand-alone object and will undoubtedly lead to establishing the new *Project role* concept in the future development of the model. Nevertheless, this fact is already out of the border of interest in our simple example.

reducible to just the area of real-time processing. There is also a need for recognising the states and transitions among them in the area of data processing. The best proof of this idea is the concept of the object life cycle itself – once we think about the objects generally (i.e. in terms of their classes), then we have to strongly distinguish between the class and its instance. In the case of the object life this requires determining those points in the life of all objects of the same class, which we will be able to identify, and which it is necessary to identify in order to describe the synchronisation of the object life with life cycles of other objects. Such points of the object life are its states. So each object instance lives its own life while the lives of all instances of the same class are described by the common life cycle.

Diagram constructs:

Construct	Meaning
	Object state.
	initial pseudo-state. Beginning of the life cycle.
	Final pseudo-state. End of the life cycle.
	Transition from one state to the other.

Each described life cycle has to correspond to the particular object class in the Class Diagram. In such way, the State Chart specifies the general mechanism of the life of all possible instances of the given class. Described states and transitions among them consequently correspond to the attributes and methods of the class. In fact life cycle states represent the specific attribute of the class (this attribute is not present in the class description but it exists by the definition – it is necessary to distinguish from among particular

states/values of this “hidden” attribute). Each transition between life cycle states then represents the use of the particular class method.

Example at Figure 24 shows the life cycle of the object Teacher from the conceptual model at Figure 23. The model recognizes basic states (life stages) of every object of the given class and should also fully respect its context described in the conceptual model. In this model, one can see not all meanings of the relationships of the class Teacher to the classes Student, Course and Class from the conceptual model are respected in the transitions between particular states. It means this life cycle is not fully consistent with the conceptual model and should be improved for that purpose. Particularly, no transitions related to the teacher’s roles “Course guarantor” is present in the model and also some possible life states related to the various types of the supervision together with the corresponding transitions are missing in the model.

The topic of essential relationships of various related models is discussed in deeper detail in the *Chapter 5 Consistency of business system models*.

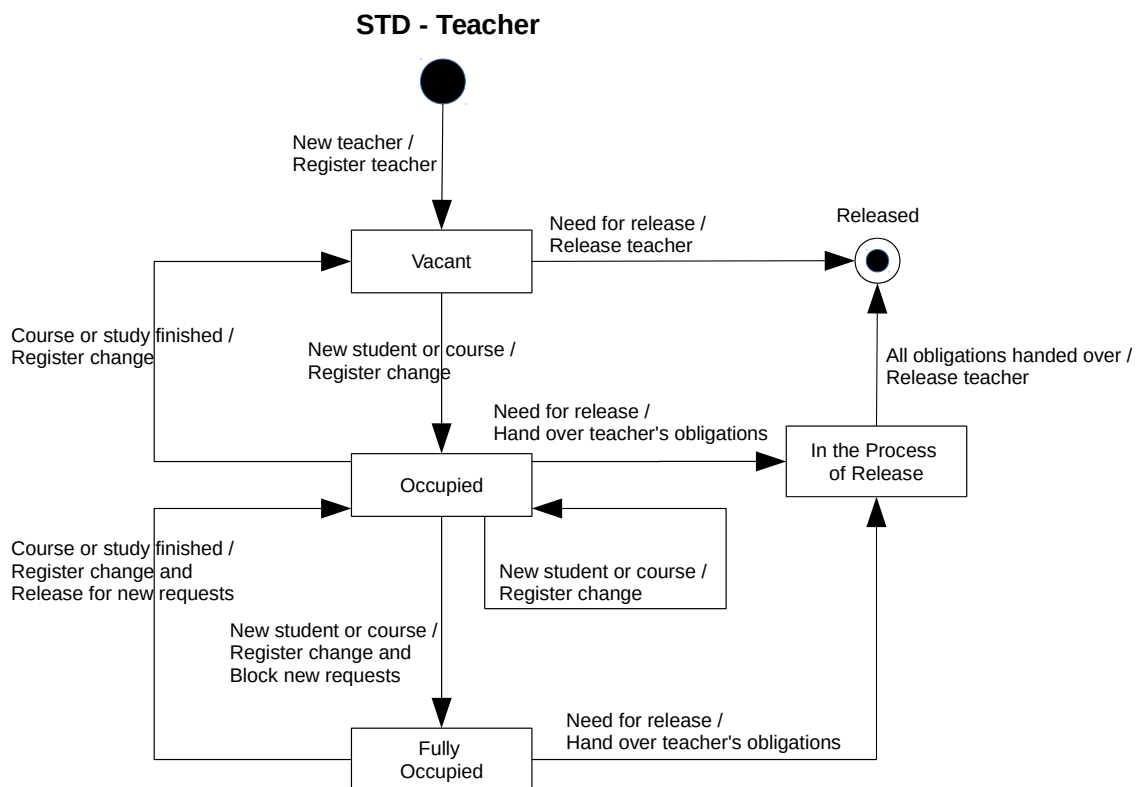


Figure 24: Example life cycle of the class Teacher from the conceptual model at Figure 23

Source: Author

While the method of the transition realisation corresponds to the specific method of the given class, the reason for the transition corresponds to the specific event (external influence) which causes the transition. The concept of events, as a common concept existing in both main points of view on the Real World dynamics, allows linking of the description of object life cycles with the description of business processes (see *Chapter 3.3 Process memory and MMABP process abstraction levels*).

Although the object life cycle as well as the business process are both process descriptions (description of the dynamics), there is a dramatic difference between them in the meaning of the “process” concept. During the Real World modelling it is necessary to clearly distinguish between the “business process” and “process in general” concepts. On one hand it is necessary to model just the Real World processes and not the infrastructure processes, which are models themselves (i.e. “software processes”, organisational procedures, performance of IS, etc.). On the other hand, the model of objects also describes the behaviour – in the form of entity life algorithms (ordering of methods). Such behaviour is seen from the point of view of objects and their relationships. It says nothing about the superior reasons for it. So, the behaviour of the objects should be regarded as the structural aspect of the real world, i.e. something completely different from the business process. In the form of the process the Object life cycle thus describes no more than the set of rules which are given by the substance of the business and should be respected by all objects of the given class (business rules). Unlike the process of the object life (object life cycle) representing just the internal logic of the object behaviour, the business process always represents some external business goal or any other form of superior reasons for behaviour. Business process thus expresses the intentional combination of object actions.

Chapter 5 Consistency of business system models

This chapter outlines the sense, and the methodical way, of ensuring the mutual consistency of different kinds of models which the Business System Model consists of. In more detail, the chapter analyses the basic kinds of model coherency, introducing the two main criteria of completeness and correctness of models, together with the concept of the structural coherency of models.

5.1 Coherency of models

Regarding the coherency of models, let us introduce two basic criteria:

- *completeness* of models,
- *correctness* of models.

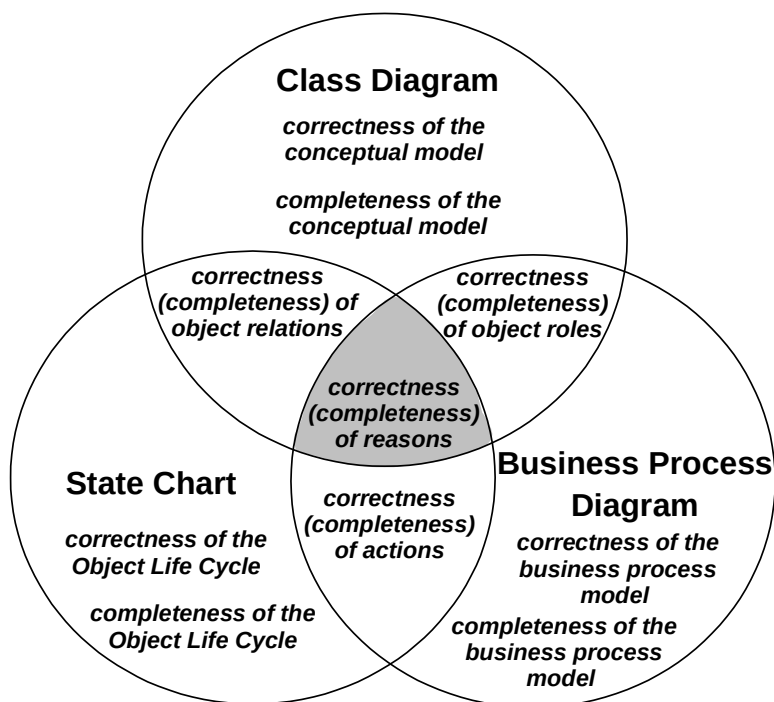


Figure 25: Criteria of completeness and correctness in diagrams

Source: Author

Figure 25 illustrates, completeness and correctness are mutually interconnected. On the level of the particular diagrams, each criterion has a specific meaning. Nevertheless, in the intersections of particular diagrams, as well as in the intersections of all three diagrams, both criteria convene together. More exactly: the correctness of the models has the form of completeness of the superior general concepts (relations, roles, actions, and reasons) in them.

The specific kind of model coherency is the coherency of the main types of structures, which occur in all viewpoints in several forms. We call this kind of model coherency *structural coherency*. It is described in the following chapter.

Completeness

Completeness has specific form in each model (diagram):

Completeness of the conceptual model generally follows from the theory of conceptual modelling, where the basic rules for this criterion are defined. For instance, one of the main rules is: “There must be at least one path between any two classes in the Class Diagram.”

Completeness of the business process model generally follows from the theory of business processes re-engineering and modelling, where the content of this concept in the field of business processes is defined. For instance, some of the main rules are: “There must be a business process model described for each specified product.” or “Each recognised event must be used in at least one business process model as a reason for some action.” (By the way: this rule defines the objective need for the breakdown of the processes – we need to breakdown the processes until we place all the events).

Completeness of the Object Life Cycles is expressed by the simple rule that the “Object Life Cycle must cover the whole life of the object.” As a realisation of this rule, the methodology defines three mandatory types of object methods (stereotypes): constructor, destructor, and transformer. The purpose is to ensure the completeness of the whole object life in the description.

Correctness

Correctness of the conceptual model is defined as follows: “Each object class must correspond to real and existing objects. Any relationship to other object class(es) must model the existing possible relationship. The described object classes and their relationships must be valid for all possible instances of each object class.”

Correctness of the business process model is defined as follows: “The business process must fulfil the main process goal. Described process actions, their succession, inputs, outputs and other attributes must be valid for all possible instances of the process.”

Correctness of Object Life Cycles is defined as follows: “The Object Life Cycle must correspond to the real and objective actions and their successions in the life of the object. The Object Life Cycle must be valid for all possible instances of the object class.”

Correctness and completeness

Correctness (completeness) of object relations considers the relationships between State Chart and Class Diagram and is defined as follows: “Each association belonging to the class in the Class Diagram must correspond to some method specified in the object life cycle (State Chart) of this class as an attribute of the state transition, and vice versa.”

Correctness (completeness) of object roles considers the relationships between the Class Diagram and the Business Process Diagram and is defined as follows: “Each object class must be present in some Business Process as an input, or Output Set, Actor or any other external factor, and vice versa.”

Correctness (completeness) of actions considers the relationships between the Business Process Diagram and the State Chart and is defined as follows: “Each action in each business process must correspond to at least one transition between states in at least one object life cycle, and vice versa.”

Correctness (completeness) of reasons considers the relationships among all three diagrams and is defined as follows: “Each event used in each Object Life Cycle as a reason for the state transition should correspond to the same event used in at least one Business Process as a reason for the process activity, and vice versa.”

5.2 Structural coherency

The roots of the idea of structural coherency are in ideas of Michael Jackson, formulated in his “JSP” method ([11]).

According to the author, the fundamental idea of JSP was: the *program structure should be dictated by the structure of its input and output data streams* ([9]). If one of the sequential files processed by the program consisted of customer groups, each group consisting of a customer record followed by some number of order records, each of which is either a simple order or an urgent order, then the program should have the same structure: it should have a program part that processes the file, with a subpart to process each customer group; and that subpart should itself have one subpart that processes the customer record, and so on.

The execution sequence of the parts should mirror the sequence of records and record groups in the file. The program parts could be very small and not, in general, separately compiled.

The resulting structure can be represented in a JSP structure diagram, as in Figure 26:

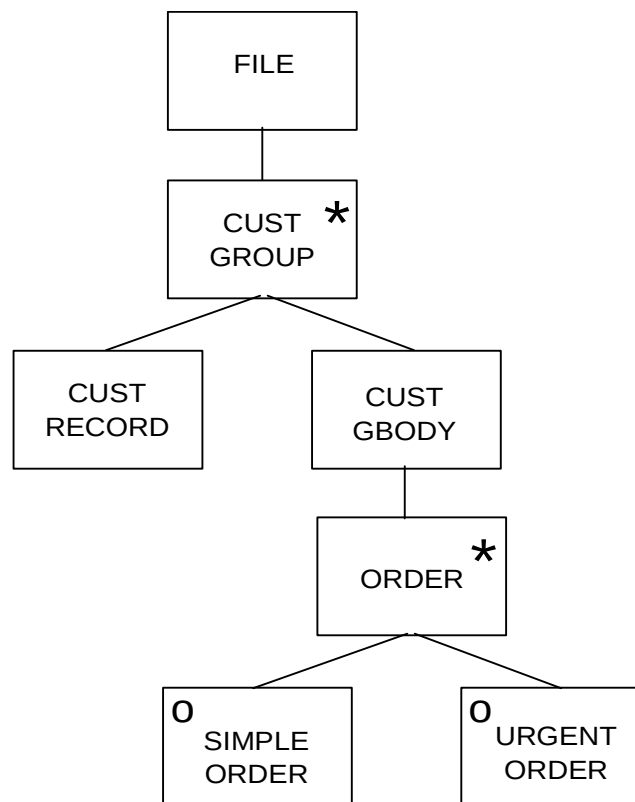


Figure 26: Structure of a file and of a program
Source: [11].

The structure is simultaneously the structure of the file and the structure of a program to process the file. As a data structure it may be verbalised like this:

“The File consists of zero or more Customer Groups. Each Customer Group consists of a Customer Record followed by a Customer Group Body. Each Customer Group Body consists of zero or more Orders. Each Order is either a Simple Order or an Urgent Order.” ([11]).

Based on the above stated idea Jackson proposed the process of designing a program which consists of the following steps:

1. Draw data structures for program input(s) and output(s).
2. Form the program structure based on the data structures from the previous step.
3. List and allocate operations to the program structure.

4. Create the elaborated program structure with operations and conditions added to the basic program structure.
5. Translate the structure diagram into the structure text or program code.

The result of applying JSP is a program that reflects the problem structure as expressed in a model of its inputs and outputs (see Figure 27). If changes to the program are required that only affect local components, the changes can be easily made to the corresponding program components. A program's structural integrity – its correspondence with the problem's structure – is the main way we can reduce errors and costs in software maintenance.

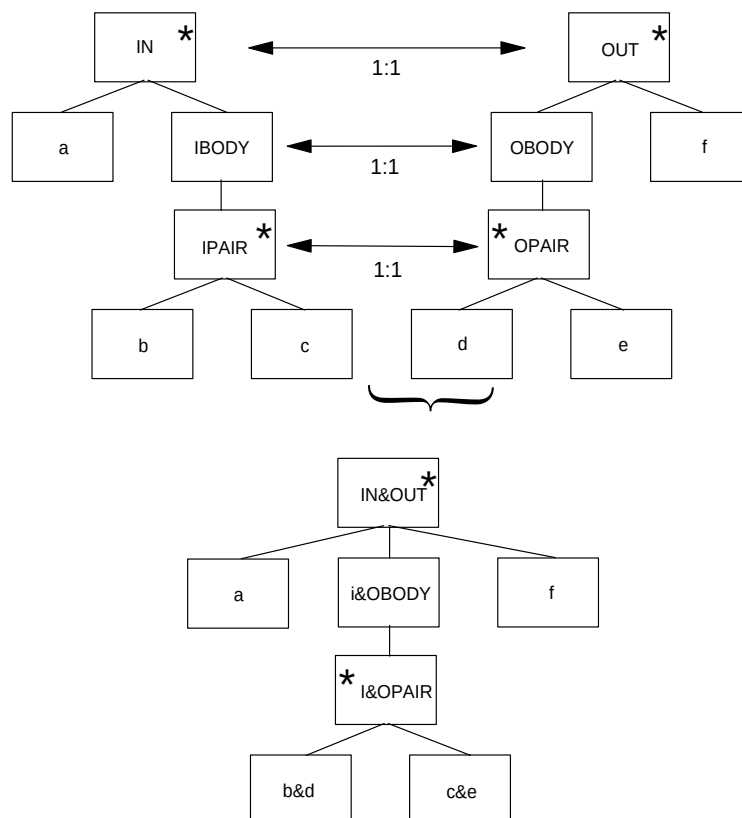


Figure 27: Two file structures and a program structure
Source: [11].

The crucial moment of the design process is the first step and the transition from the first to the second step. In fact, *the root of the problem is solved by merging data structures together* – it requires making a set of crucial decisions about the correspondences of the particular data structures parts and their merging into the resulting structure (which is, in fact, the structure of the transformation process from the input structure(s) to the output one(s)). Therefore, Jackson determined the set of rules for merging structures together. In addition to this set of rules, he defined the concept of the “structure clash”:

if there are two non-corresponding components of the corresponding iterations, and if it is not possible to merge them as a sequence, or as a selection, nor to express the first component as an iteration of the second one (and vice versa), then there is a structure clash existing between both structures (see Figure 28).

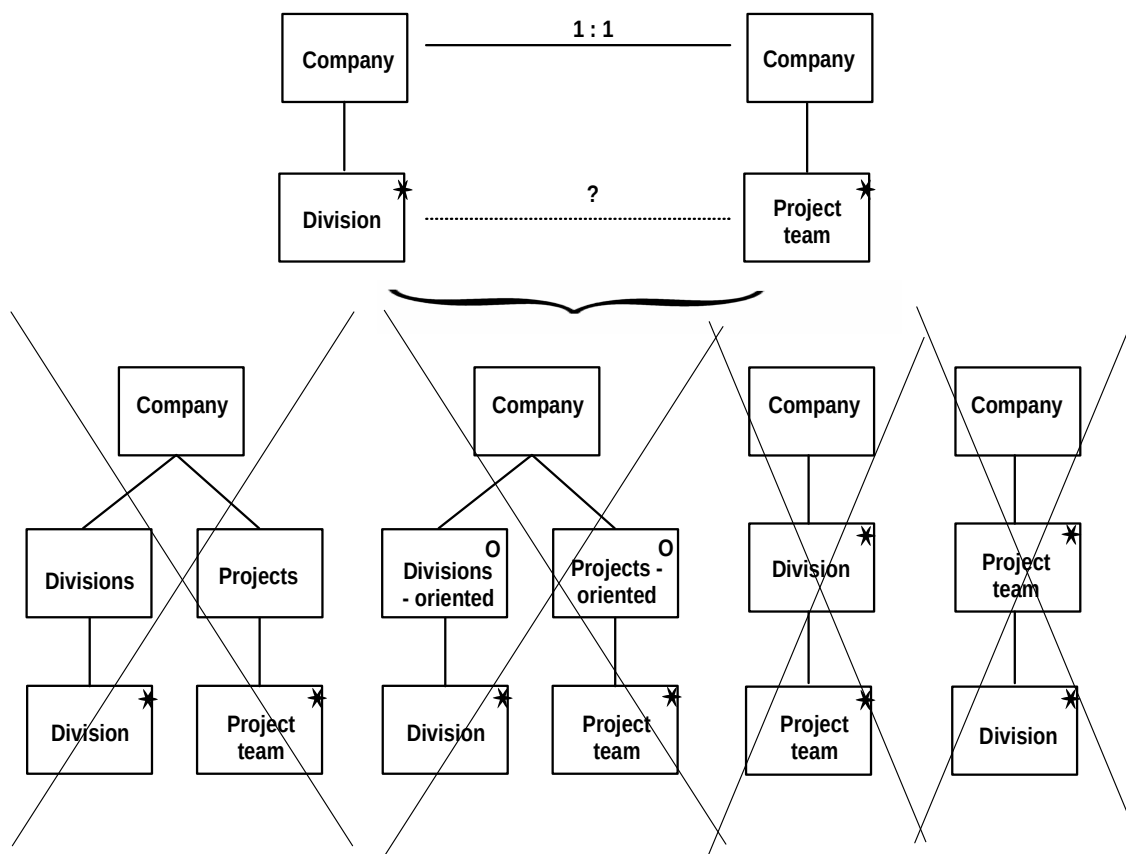


Figure 28: Structure clash

Source: Author

The structure clash means that it is *impossible* to express the substance of the problem as a *simple structure*. The process of solving the problem, consequently, requires several *parallel* solution processes, each targeted on separate, and relatively independent, parts of the whole problem (sub-problem). For instance, in the example given in Figure 28, the solution (i.e. transformation of the company from the division-oriented organisation to the project-oriented one), requires breaking the company organisation up at first, and then building the new organisation which is harmonised with the project management requirements. This is because the division organisations, and the project organisation, are mutually independent in so far as it is not possible to make any compromise, or to subordinate one structure to the second one.

Thus the *structure clash* (from Jackson's theory) is the precise technical *definition of natural parallelism* in the process, stemming from the nature of the problem itself and, therefore, substantially present.

Structure Diagram used for the description of the objects' life cycles on the left side of Figure 29 is not the regular UML diagram. UML prefers an unstructured way of describing the context as it follows from historical circumstances. Right side of the figure illustrates the same life cycle described in the regular UML diagram – State Chart. It can be easily discerned that the difference between these two styles of precise description reflect the basic differences between the structured and unstructured views. In the structured view, there is the need for creating abstract higher units (such as “Living” and “Filling”). These units serve as a way of understanding the problem. Such possibility is missing in the case of the State Diagram. On the other hand, the State Diagram allows for the reverse point of view of the object life cycle – in terms of states and the transitions between them. Such a view is very close to the position of the real attendee of the business (i.e. real world behaviour) who usually sees the process in detail. A very serious problem is that this unstructured description often leads to the creation of subsidiary abstract concepts which have nothing to do with the real world (i.e. concepts which are not “conceptual”). In our example there is the problem with the presumption that the same “Exemption” event occurs twice, which is impossible in the real world. The reason for this is the impossibility of expressing the necessary combination of actions connected with this event in the given situation (object state).

The State Chart is not primarily intended for a description of the life cycle; its roots are in the area of the State Machines Theory, and it is closely connected with the concept of the so called “real-time processing”. However, this concept of the state machine is general as it is not substantially reducible to just the area of real-time processing. Also, in the area of data processing there is a need for recognising states and transitions among them. The best proof of this idea is the concept of the object life cycle itself – once we think about the objects generally (i.e. in terms of their classes), then we have to strongly distinguish between the class and its instance. In the case of the object life this requires us to determine those points in the life of all objects of the same class, which we will be able to identify, and which it is necessary to identify in order to describe the synchronisation of the object life with the life cycles of other objects. Such aspects of the object life are its states. So each object instance lives its own life while the lives of all instances of the same class are described by the common life cycle.

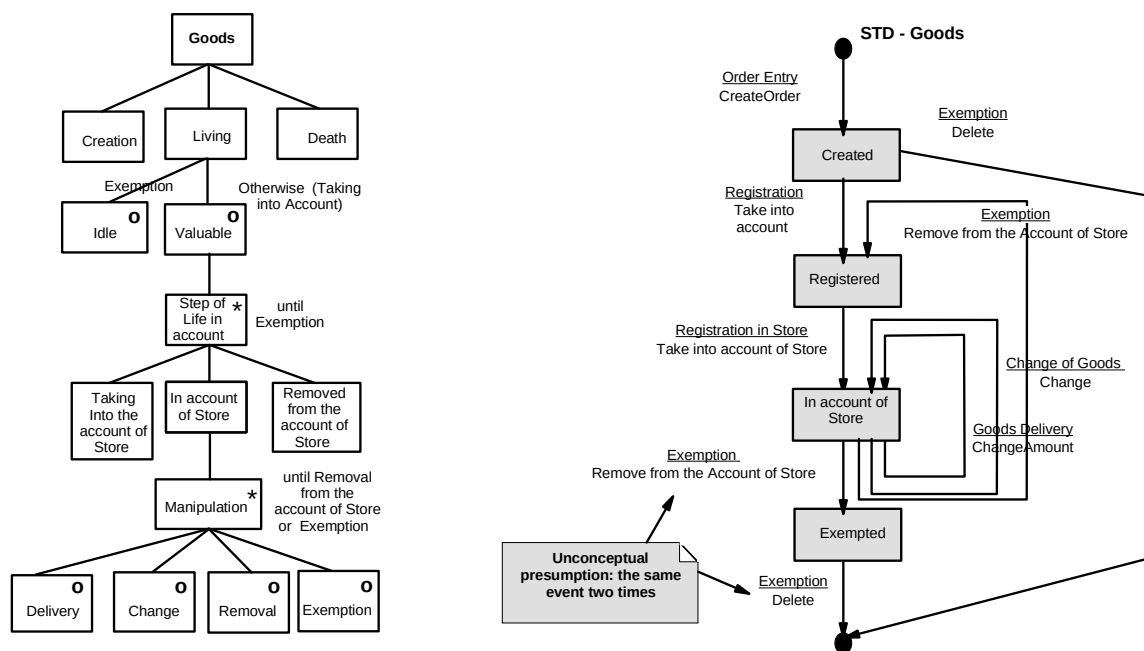


Figure 29: Structure diagram versus State Chart

Source: Author

Let us revisit some limitations of the state-oriented description:

- An unstructured view of the process requires the need for the additional reader’s abstraction in order to recognise the structures.

- For a description of the generalised processes it is necessary to use the hierarchy of diagrams (compound states). When we describe the life-cycle of the object class, this necessity is warranted because such a process is generalised by definition.

On the other hand, the main limitation of the operation-oriented description is the fundamental need for the reader's abstraction – the reader needs to generalise sets of operations in order to recognise the basic structure types (iteration, sequence, and selection). Such a need does not occur in the state-oriented view of the life-cycle, where the description strictly follows particular state transitions. Nevertheless, this abstraction is necessary for recognising deeper – structural – consequences of models.

5.3 Mutual consistency of object life cycles

Jackson's rules for merging structures, as described above, allow us to take the structure as a common denominator of both the data and the process and use of this structure as the basis for mapping deeper conjunctions among data structures and processes.

Moreover, there are some other general analogies which could be useful for utilising Jackson's ideas for reflecting the natural consequences in Real World Models, which follow on from the nature of the relationships among the Real World Objects. In the following text I call them *structural consequences*.

The main, and the most important, general analogies, mentioned above, are:

- The *sequence* type of structure is an analogy to the *aggregation* type of hierarchy, while the *selection* type of structure is an analogy to the *generalisation* type of hierarchy. In this connection, it is necessary to consider that the *iteration* type of structure is just a special case of the sequence (where all its parts are of the same structure), hence it is an analogy to the *aggregation*.
- The *cardinality* of the relationships among objects is an analogy to the *aggregation* (as the aggregation reflects the quantity and says nothing about the quality), whilst the *optionality* of the relationship is an analogy to the *generalisation* (as the generalisation reflects the quality and says nothing about the quantity (including the ordering)).
- Similarly, the generalisation (inheritance) type of relationship in the class diagram should be reflected by some kind of selection, whilst the aggregation (composition) should be reflected by some kind of sequence/iteration, with all consequences following from it.

Jackson's theory does not only describe the rules for merging structures together. It also leads to the important idea that structural coherency is the crucial point for modelling the basic relationships between the static dimension of the real world (what it consists of), and its dynamic dimension (how it is doing). Each point of view of the real world, including the conceptual model, has these two dimensions. In the conceptual model of the real world the static dimension is modelled by the conceptual object classes and their relationships, whilst the dynamics of them is modelled by their life cycles.

We may conclude from previous paragraphs that we can formalise basic rules for the structural consistency of objects in the conceptual model as follows:

- Each association between two object classes must be reflected by a specific operation in each class life cycle.
- The cardinality of the association must be reflected by a corresponding type of structure in the life cycle of the opposite class: cardinality 1:n by the iteration of parts, cardinality 1:1 by the single part of the structure.
- The optionality of the association must be reflected by a corresponding selection structure in the life cycle of the opposite class.
- Each generalisation of the class must be reflected by a corresponding selection structure in its life cycle.
- Each aggregation association between classes must be reflected by a corresponding iteration structure in the life cycle of the aggregating class (container/composite class).

Figure 30 illustrates some examples of structural coherences in the conceptual model. The Class Diagram represents the static contextual view of reality, while the object life cycle describes the "internal dynamics" of the class. The internal dynamics of the class should be subordinated to the context (i.e. substantial relationships to the other classes); therefore, each class contains a specific operation (method) for each association (it is obvious that some associations to other classes are missing in this example). The life cycle determines the placement of each particular operation in the overall life history of the object – the internal context of the operation. The internal context must be consistent with the external one, which follows from the relationships described between classes in the Class Diagram (associations to other classes, generalisations, etc.). Dashed arrows indicate the basic consequences of the described associations and their cardinalities in the life cycles of both classes:

- Optionality of the association (goods may not to be ordered at all) is reflected by the existence of the possibility that the whole substructure, representing the ordering of goods, may be idle in the Goods life cycle. Also, the fundamental conditionality of the delivery is a reflection of this fact.
- Multiplicity of the association (one Order may contain several items) is reflected by the iteration of the “Filling” structure in the Order life history, which expresses the fundamental fact that the order may be created, fulfilled by several supplies, or changed several times; separately for each ordered item.

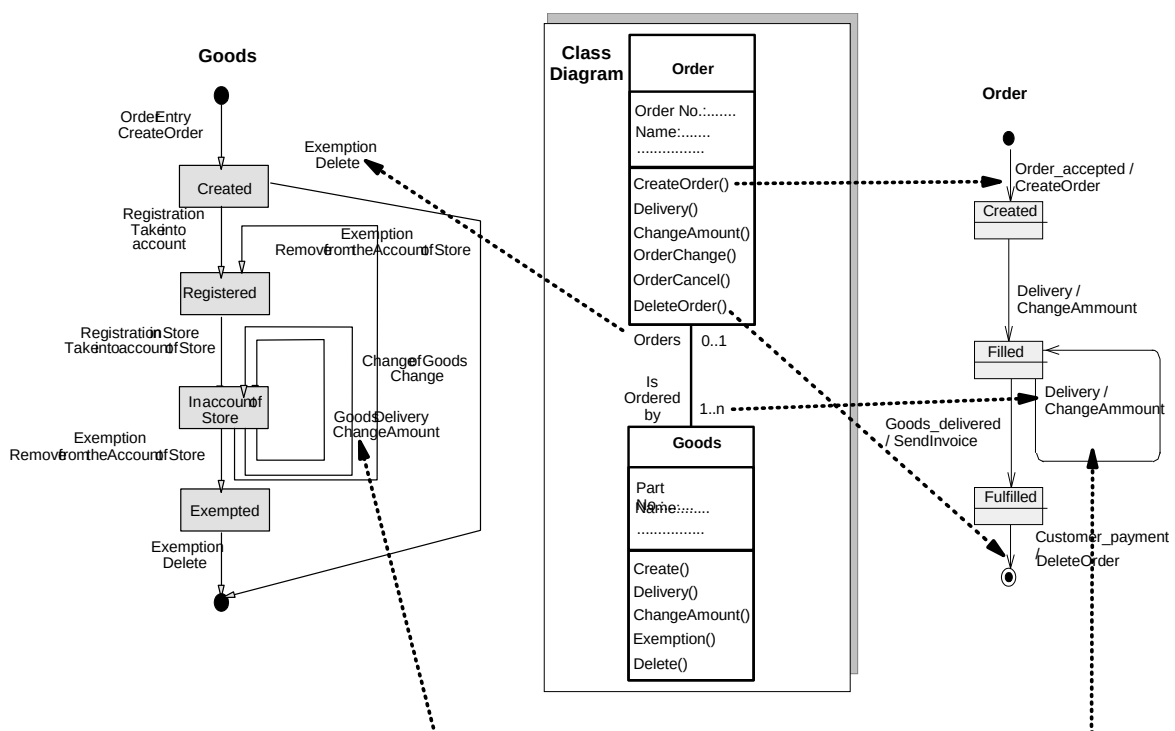


Figure 30: Structural coherency of objects and their life cycles

Source: Author

Knowledge of the structural consequences helps the analyst to improve the Real World Models concerning their mutual consistency, as well as, their relative completeness (as completeness is a main part of the problem of consistency).

5.4 Linking processes with objects

Process model and Class model have to be also mutually linked. This linkage represents the logical relationships between these two basic views on the real system:

- view on the Real World as a structure of objects and their relationships which are relatively stable – the Class model view
- view on the Real World as a structure of mutually related actions which are processing inputs into the outputs in the name of the process goal – the (Business) Process model view.

Since these two views are just different views on the same Real World they have to correspond together – to be mutually consistent. This means that business objects which are modelled in the Class model have to be present in the Process model in various forms as actors, inputs/outputs, organisation units and other external aspects, for example.

The sense of this double-vision of the Real World is in fact that these two views are different. Two different views on the same system are allowing the spatial effect – the new information as a synergistic effect.

As these both views are different views on the same Real World it is necessary to know what their common meaning is – how the elements of one view are corresponding to the elements of the second one. Taking the level of difference between these two views into account supports that such correspondence is not simple. One object from the class diagram typically occurs in a number of processes, and one process typically combines a number of objects in a number of roles. Moreover, one object can even be found in the same process in several different roles (as an actor and information input in the same time, for example), and vice versa.

The following set of rules follows from the above described facts:

- (1) *Every objects class of the Class model has to be present in the Process model in at least one of its inputs, outputs, and/or as an actor or other external element(s),*
- (2) *Every input, output, actor or other external element of the process has to be present in the Class model as a class, or as an association, or as a combination of both.*

The above described rules are relevant only for one side of the reality – the aspects of the existence. This side of the reality consists of objects and artefacts and their general (substantial, static) relationships. However, the other side of the reality is also present in both models. This side consists of the actions, states, and their (time) relationships. While in the process view this “behavioural” aspect of the Real World is present naturally, in the object

view it is not so simple to perceive it. This is exactly the reason for the description of the object life cycle.

Outline of some basic kinds of links between the business process and business objects description using the object life cycles can be found in Figure 31 and Figure 32. The figures show the correspondence of both mentioned views on the Real World. Both views describe events and actions, but from different viewpoints – as an evolution of the object during its life and as a succession of the business process actions. It allows seeing which aspects of the reality are invisible in one viewpoint from the other viewpoint.

The following set of rules completes the previously stated rules with the aspects of the object life cycle description and their manifestation in the relationships between the process and the objects models:

- (1) *for each object class from the Class model at least three methods have to be specified:*
 - a. *the constructor (it creates the instance of the class),*
 - b. *the destructor (it deletes the instance of the class),*
 - c. *the transformer (it changes the attributes of the class),*
- (2) *for each attribute of the object class the method which initiates the value of this attribute has to be specified, as well as the method which changes the value of this attribute,*
- (3) *for each association of the object classes the corresponding method which realises this association has to be specified,*
- (4) *for each non-primitive object class from the Class model the State Chart describing the life cycle of the objects of this class has to be created,.*
- (5) *In the State Chart describing the life cycle all possible (allowed) transitions among all states should be described. Every transition has to specify the causing event and the method used for the transition realisation,*
- (6) *the State Chart describing the life cycle of the objects of particular class has to use all methods of this class in the transition descriptions. Transition description does not use the method which is not specified in the Class Diagram,*
- (7) *Every event used in the description of the transition has to correspond to the event specified in the description of some business process(es).*

The topic of relationships between the model of business processes and the model of business objects through the object life cycles is also discussed in the following text.

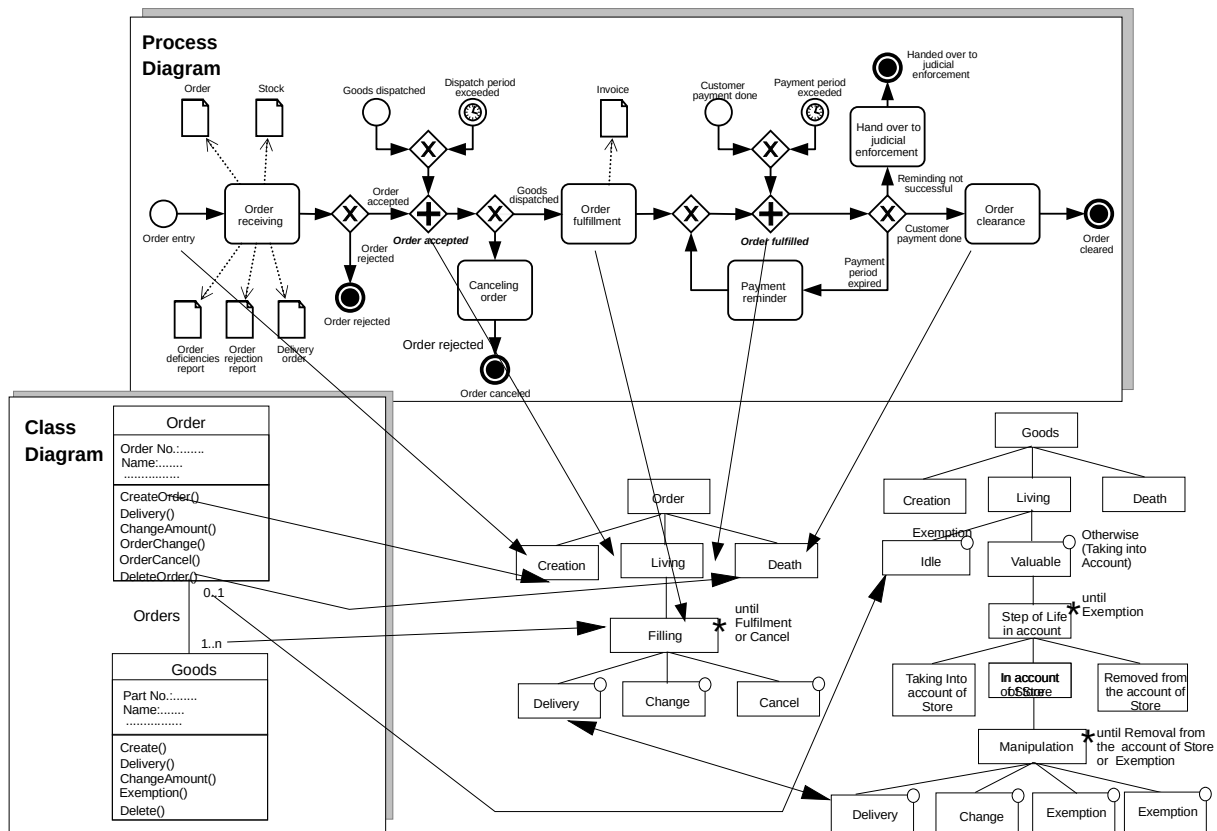


Figure 31: Example of the coherency of models

Source: Author

Figure 31 illustrates how the process model explains the dependencies between objects and their life cycles giving them a deeper meaning. This explanation is based on the perception of object actions in terms of the reasons for them – events and process states. Objects are playing the roles of attendees or victims (subjects) of processes. For completeness it is necessary to recognise the fact that one object typically occurs in more processes, as well as, one process normally combining the attendance of more objects. The orthogonality of those two points of view is also typical and substantial – it lends sense to this coupling. Structure and behaviour is the analogy of the two basic dimensions of the real world – space and time.

As the Structure diagram is an algorithmic-oriented way of description of the life cycle, its mapping to the process activities is direct and easily understandable. On the other hand, the process states are not so directly visible in the life cycle. They are related to the transitions between life cycle actions. Figure 32 shows the same example using the UML State Chart. Besides the fact that the state-oriented way of description of the life cycle is much closer to the natural understanding of the lifetime (states / life stages), life cycle states can be directly mapped on the process states which makes this consistency relationship easily understandable. Nevertheless, the example also shows the limitation of this “unstructured” description. Unlike in the Structure Diagram, where the types of structures are explicitly expressed and can be therefore easily mapped on the same structures in other models in terms the application of Jackson’s theory (see the previous chapter), in the State Chart the types of structures are hidden in the relationships of different transitions and their directions. Iteration is related to the possibility to repeat the same state by the follow-up transitions. Similarly, selection is related to the fact that there are more output transitions from the same state.

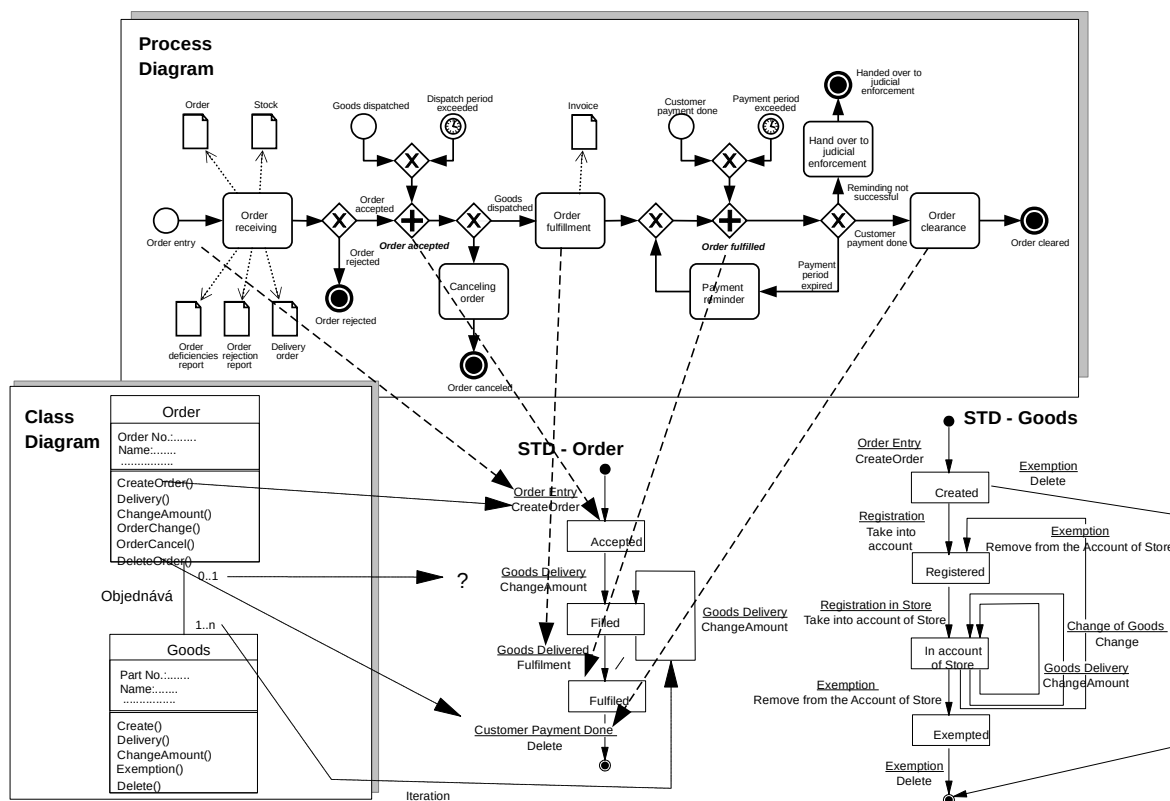


Figure 32: Relations between process and object models through the object life cycles

Source: Author

Even the specified consistency rules work together in mutual coherency. This means that a number of additional second- and third-level consistency rules following on from the combination of basic rules should be considered. For example: We suppose that each event specified in the Object Life Cycle is used in some Business Process(es) (the rule for correctness (completeness) of reasons); and at the same time we require each state transition in the object life cycle to correspond to some association with another object class (the rule for correctness (completeness) of object relations). From this combination of rules it follows that we suppose that each event causes some business action (as it is defined in the business process model) and that it causes the state transition of some object (as it is defined in the object life cycle), and that it fulfils the link to some other object at the same time. In fact, this means that each business process activity has logical consequences in the mutual behaviour of objects (and vice versa¹⁶).

¹⁶ in fact, here we deal with the famous “chicken-and-egg dilemma”; deciding whether the mutual behaviour of objects is the consequence of business process activities or whether business process activities are, rather, given by the actor’s behaviour. This really philosophical problem is connected with the two basic phenomena that determine the business system: the logic of the business system (modality and causality) usually called *business rules*, and the intentions of business actors usually called *business goals*.

Chapter 6 Role of the information system in an organization

In this chapter, we explain the role of the information system in the process driven organization and also the relationship between information modelling of organization and the development of its information system. The final chapter then follows this topic aiming on the broader consequences of the integration of IT and business in terms of so-called *Digital Transformation* and related phenomenon: a process-driven evolution of the organization.

6.1 information system of a process driven organization

As explained in the introductory chapter, the main reason for managing the organization and its evolution by processes is the need for the flexibility towards the evolving technology with keeping the control over the organization's evolution at the same time. Simply speaking, we have to be able to operatively manage the relationships of particular tasks in the organization defined as business processes instead of "binding" them in the organization structure. This freeing of the business processes from the organization structure allows keeping the evolution of the process under the control, to flexibly change its structure towards the new possibilities given by the technology development. The behaviour of the organization is then flexible. Indeed, all supporting infrastructures of such a flexible process system has to have the same level of flexibility. To achieve the needed flexibility of the information system, we need to control the "behaviour" of the information system by a standalone component, which allows to control the relationships between particular functions of the information system by means of the process definitions.

Figure 33 shows the needed structure of the information system of a process driven organization. The behaviour of the information system in the process driven organization is controlled by its key component: so-called *Workflow Management System*. This component contains the model of the processes and call particular functional components of the information system according to the process definitions. In the same way, this component also uses the database that, besides the essential data processed by the functional components, contains also the data about processes and for their support (exactly speaking for the support of their control). Compared to the traditional conception of the enterprise information system, where the relationships between particular components are unchangeably bound in the structure of the system, this conception allows to operatively change the behaviour of the system exactly according to the actual current need of the running process.

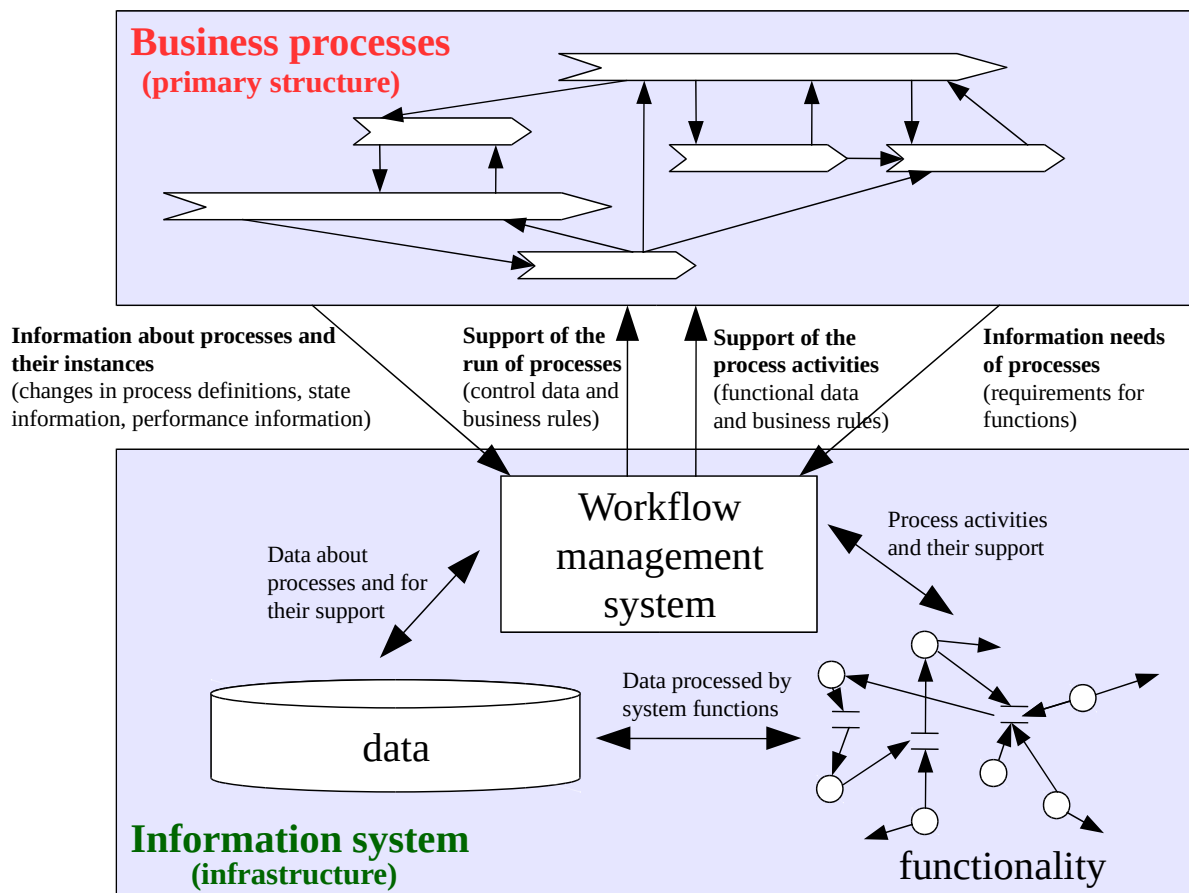


Figure 33: information system of a process driven organization

Source: Author

Figure 34 show a general structure of the *Workflow Management System* as it is defined by the Workflow Management Coalition standards. The central component of such a system is so-called *Workflow Engine*. It controls the behaviour of the system by calling the components of the system (the *client* and *invoked applications* in the figure) according to the definitions of the currently running processes. It also allows to work with the definitions of processes: to create and operatively change the definitions and consequently, to accommodate the behaviour of the system to the current need. Other functions of the *Workflow Management System* are the administration and monitoring of the process run and also enabling the possible collaboration with other workflow engines if needed. The need for the above-mentioned functions of the *Workflow Management System* causes a strong relationship to the existing standards that allow building the information system as an open system that is permanently prepared for various and currently yet unknown improvements that the future development of the technology could bring. That is a good illustration of the fact that standardization is one of the crucial aspects of the process-driven management of an organization.

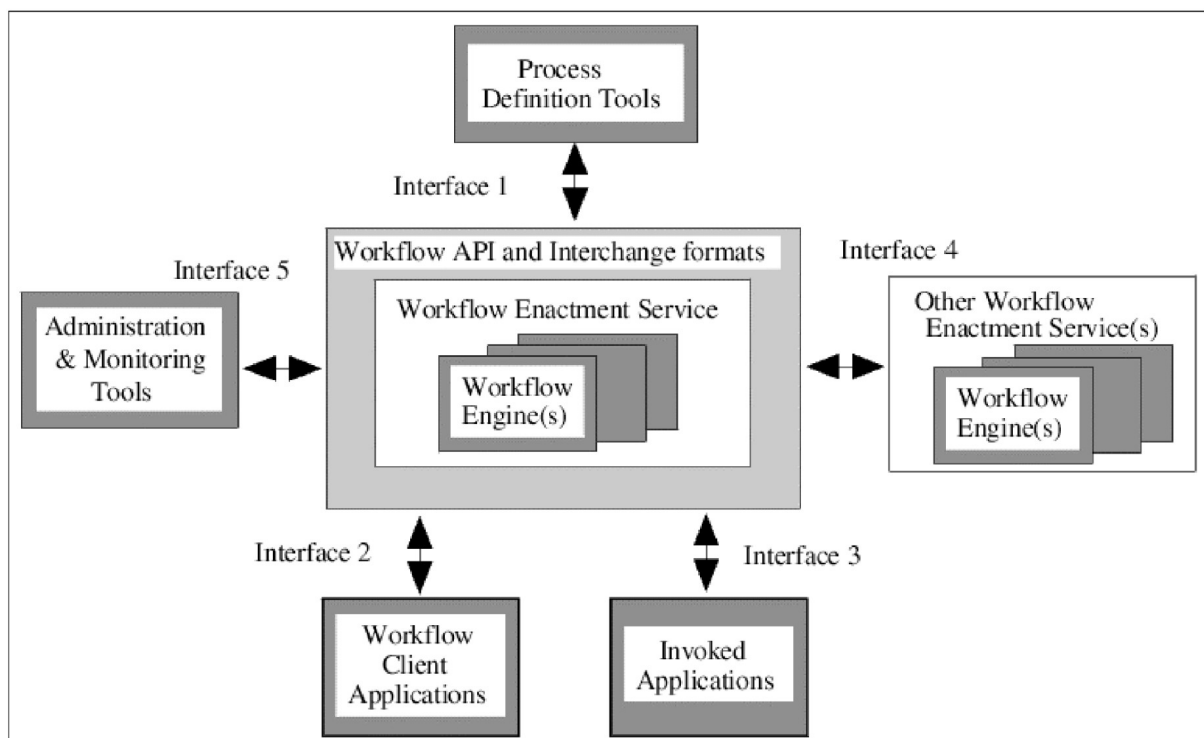


Figure 34: Structure of the Workflow Management System

Source: Workflow Management Coalition

6.2 information system functionality and Data Flow Diagram

To specify the content of the information system it is necessary to complete the models of business objects and business processes with the functional model of the information system in the form of a data flow diagram (see Figure 2 in the introduction chapter).

Functionality of an information system represents the substance of its work - that is, which business activities the information system supports and how (which data / information it provides with). In terms of the principles of modelling, the functionality of an information system is also a reflection of the real system, a reflection of real processes which are performed in order to insure the will to the creation, elimination, or support, etc. of real objects and their essential relations. In this sense, the functional model of information system should be derived from the model of real processes and model objects.

However, in the information system itself - as part of a real system - the procedural and structural elements of reality manifest themselves in a specific way: as the data / information on real interrelated events, the knowledge of which is embedded in the intrinsic properties of an information system. The information system receives the information on real event in the form of its input (data flow) to combine it with other known events in order to derive information from their relationship. Combining information about different time-independent (asynchronous), but factually related phenomena requires storing the information (ie remembering the event and waiting for the next future, factually related, events). To do this the information system must "know" the reality. More precisely, the system must be able to assume primary (essential) continuity of basic (essential) real events, these expected relations must be "coded" in its structure.

Figure 2 describes the elements in a functional model which coincides with the model objects. They are both attributes of individual objects in the information system implemented in the form of data elements and structures (which reflect the interaction of objects), both methods of objects implemented in the form of information system operations and their structures (also reflecting the relationship of objects). Figure also describes the elements in which the functional model overlaps with the model of processes. Here are the events to which particular processes are reacting, realized in the information system in the form of data flows and structures (which reflect the interaction of processes - their communication), the activity of the processes implemented in the form of operations and information system functions and structures (reflecting the communication between processes as well).

The figure also shows common intersection of all three models - the common denominator of all bilateral common elements are the data structures (reflecting the attributes of objects and events and their essential relations), and also the essential regularities, reflecting the essential continuity of activities and events (the life cycles of objects).

The actual content of the IS function is a kind of specific (specialized) model of the real system in both its dimensions. Then, however, the functional content of the information system must be strictly distinguished from the form of its implementation (which is given generally by the technology environment and particularly by the implementation environment). Thus the function model of the IS in terms of information system remains the model of content - technological, and implementation models will be derived from it. Some authors have therefore called it the essential model (e.g. E. Yourdon in his pivotal work [46]).

Data Flow Diagram

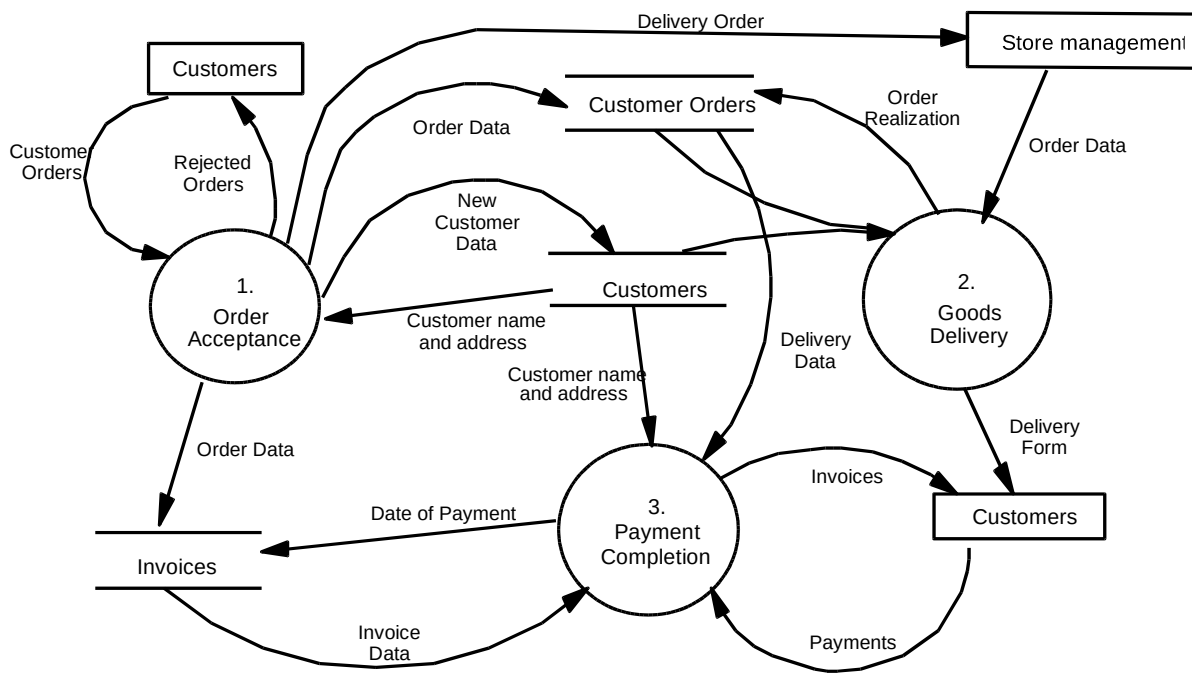
Data Flow Diagram (DFD) specifies a functional model of the information system as a set of the functions and their relationships. Function determines what processes are conducted in the information system in order to be a faithful model of supported reality. The processes of information system actually reflect the real business activities and processes. Nevertheless, functional model is not an algorithmic model (even if it describes processes). DFD describes functions which exist in the system, and substantial connections between them via the data flows. It is a static description. Described processes are processes of information system, processes which are modelling, not the business processes which are modelled. Just a static description of functions (non-procedural) is needed during the analysis to be sure that all important behavioural aspects of the reality are fully modelled by the information system. Procedural aspects of the behaviour of an information system are not the subject of analysis of business reality. Algorithmic description of the specific information system processes is done in the phase of the system design (see eg. Deployment Diagram from the UML). The functional model of the system thus represents the basic analytical terms of reference and a starting point of the system design.

Figure 35 shows the data flow diagram describing how the data flow through the information system functions from its inception (the "Terminator") to the final consumption ("Terminator"). These natural data streams (given by the nature of the real system) are interrupted on their way by saving them to the "data stores", because of the need to wait for more information on other events in order to allow its interpretation in the context of the information stored.

DFD has evolved from the so-called Activity Diagrams used in the SADT methodology [17]¹⁷. SADT (Structured Analysis and Design Technique) is a methodology, used since about the mid seventies. DFD is the most thoroughly methodically seized in the work of Edward Yourdon [46] (see also www.yourdon.com).

The DFD has following basic elements: Function, Data Flow, Data Store, and Terminator (external entity).

¹⁷ Note that the SADT Activity Diagram has (except the name) nothing common with the UML diagram of the same name, which is rather an analogy of the classic flowchart for the description of the algorithm.



Element Types

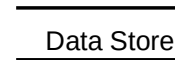
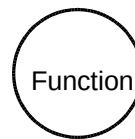
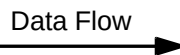
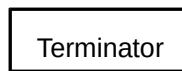


Figure 35: Data Flow Diagram

Source: Author according to [46]

Function is an essential element of the diagram. It represents the process of data processing. According to the Principle of modelling, the function is an element of the real world behaviour model. Nevertheless, it can not be considered a business process, as it uses to be often but erroneously considered. The function is a process, running in the information system, which itself is a specific model of the real (business) systems (including business processes). The function thus represents a unit of behaviour (performance) of an information system. Thus function of the IS is never related to "physical" elements of the real system behaviour - business processes 1:1, but rather M: N, since typically many business processes need to meet the same IS function to support only part of their needs. Physically, the function represents the transformation of data, which leads to the production of an output (input to output transformation).

The difference between the business process and the IS function is clearly manifested by a very popular service-orientation principle in the IS development. This principle is based on the fact that it is necessary to strongly distinguish between the elements of the IS functionality which are relatively stable, typical, and their temporal combinations which follow from the natural (business) processes and their needs that are specific and often temporary. Function as an element of the DFD thus represents an abstract content of the service. Consequently, the general criteria for the granularity of elementary functions in DFD should be principally the same as criteria for the granularity of services from the Enterprise Architecture theory.

Traditionally there are two basic kinds of processes, described in the DFD:

- *Data Process (function)* - expresses the physical transformation of data, ie the change in data representation, or change of some part of the data, ie the change in data values, the creation of new data. The main task of the function (process data) is to process (transform) the data.
- *Control Process* expresses the control algorithm (cross-time continuity) of functions in a certain part of the system. It is used to capture the real-time characteristics of the application. In contrast to the function, the task of the control process is not to produce data. In data processing systems thus control processes lose their meaning, they should not be considered in the analysis phase, but only in the design phase (see the Deployment Diagram from UML). It can be shown that, regarding the principles of object orientation, the presence of the information system processes in the analytical models is generally pointless. For these reasons Control Processes are no more addressed.

Dataflow represents an abstraction of any form of data transfer in the system and to / from it. Data flows contain data that are processed and stored in the system. Even though data flow diagram was originally developed for description of the flow of documents, materials, goods, raw materials, etc., it is not suitable to use this diagram for this purpose as it unnecessarily creates a hotbed of analytical errors. For this purpose different, more appropriate tools have been created, such as the A-graphs of the ISAC methodology (see [15]), or business process modelling tools (see [32]).

DataStore is an abstraction of any form of data storage in information systems. DataStore is a depository of data (data stored for later use). It is represented with two parallel lines. In the technical symbolism this symbol represents the break, which points out that the data storage means interrupting the flow of data over time. This fact has a far-reaching importance in process modelling (for example see the events analysis technique described in [46]).

DataStore as a "place for temporary storage of data" is used wherever there is a delayed (asynchronous) transmission of data between processes. It expresses only the fact that the data are kept (ie the flow is interrupted at the time) and says nothing about the particular form of storage (which is an issue of the IS implementation). DataStore is a secondary (passive) element in the diagram - data flows to and from must always be performed by functions.

Terminator is an object that does not belong to the described system, but its substantial surrounding area. The Terminator (beginning or end of the data flow, data source, the location and purpose of the data consumption) shows an external source or destination of data (sometimes also called an external entity - an object). It thus reflects the surrounding (real world) system with which the information system communicates.

Functionality of the system is described not only with one diagram but with the hierarchy of related diagrams. Each function can be described in greater detail as a separate chart on the lower level (i.e. more detailed one). In such a system of diagrams some elements of the interface functions are naturally repeating. This fact raises the general risk of inconsistencies models of both levels (ie situations where a detailed model describes the same interface element other than a higher-level model). This is called the consistency of diagrams hierarchy.

Consistency of DFD and other information model diagrams

MMABP consistency rules also substantiate the meaning of a functional model as an integrated part of information system models, as well as they manifest the strong need for instrumentation as UML, if it meets its responsibilities in the field of analysis (and not only the fulfilment of "user requirements" as it is popular in current application development approaches).

Consistency of DFD and object model

Class Diagram (CD) represents in terms of DFD a general view of the data base of the modelled system. It contains the data elements to store - the classes and their attributes - and a set of operations that can be carried out over such data in the form of an object's methods. DFD is on the other hand, a model that captures the system interaction with the environment, its response to external events, and records which data structures are influenced by processes in the system due to their reactions to trigger events.

The points of contact of these two diagrams are primarily data, and also, to some extent, the operations performed on the data (although not in relation 1:1).

In maintaining consistency of data between the DFD and the CD there is therefore particularly necessary to maintain the link between each DataStore and the structure of some classes and their relationships in the CD. It is appropriate to maintain a link between class methods and data flows in the DFD. As this consistency link is obviously much less tight, it can be omitted if there are technical difficulties with the modelling tool.

The following set of rules describe aspects of the relations between the model classes (CD) and data flow diagram (DFD)

- A) Each elementary DataStore in the DFD must be represented in the CD as a class, or association, or a combination of both.*
- B) Attributes of each elementary DataStore in DFD must be a data structure composed from attributes of classes which represent this DataStore in the CD.*
- C) Methods of each elementary function in the DFD must be an algorithmic structure composed from the methods of classes, which in the CD represent DataStore, associated by data flows with this function.*

Consistency of DFD and process model

Process Flow Model is a model of the real (business) process. Seen through the eyes of an information system - the processes are running across the system, the information system provides them with the information support. As a model of the real (business) process, the Process Flow Model does not address the structure of an information system but the transformation of the real inputs to outputs. It also monitors the resources consumed in this transformation. The activities that are displayed in Process Flow Model manifest themselves in the processes in DFD, although the relationship is not 1:1 but rather 0:N. The reason for this misty relationship is particularly the different focus and character of the two charts - one describes a business processes and the second the structure of information system functions. However, the activities of the business process should have its image in the DFD.

Since the process model and DFD are working with events, the same events should appear in both diagrams, or there would be some relationship among the events of the DFD and the Process Flow Model at least.

The following set of rules describes aspects of the relations between the process flow model and the data flow diagram (DFD):

D) Each elementary input DataFlow from the Terminator (ie, from outside the system) in the DFD must correspond to an event specified in (some) business process (processes) in the Process Flow Model.

E) Each state of each process in Process Flow Model must correspond to certain (some) elementary DataStore (DataStores) in the DFD, and vice versa (every elementary DataStore from the DFD must correspond to certain (some) state (states) of the process (processes) in the Process Flow Model. That is an M:N correspondence.

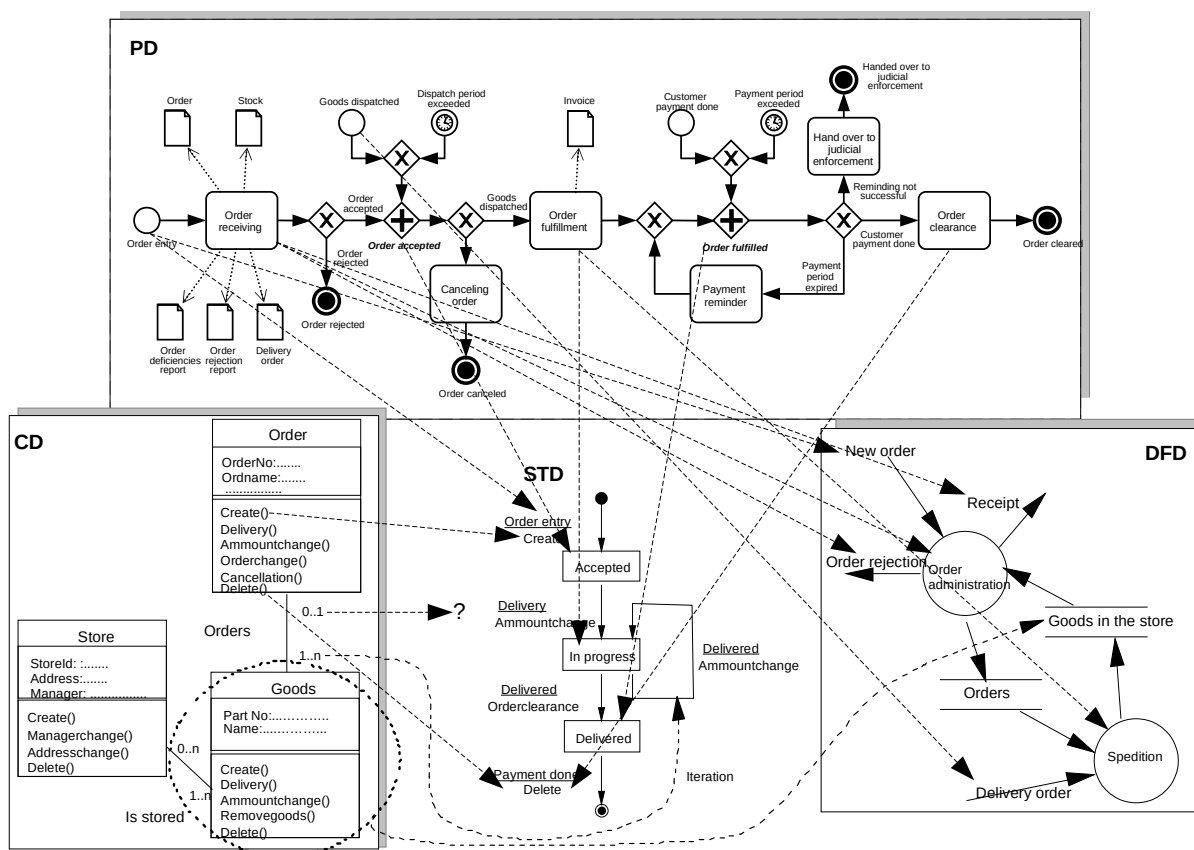


Figure 36: Consistency of the model of processes, objects, and DFD

Source: Author

Consistency of DFD and object life cycles

For the description of a general life cycle of objects of the given class, MMABP uses the UML State Chart (SC). Life cycle description enumerates the states to which the object of the class can come, and possible transitions between those states. Each transition is described by the action that occurs on an event from the external world. The transition is caused by an event that must have an image even in DFD and the process flow model. State Chart is a basic tool for describing the relationships between object and process models of reality (see Chapter 5.4 *Linking processes with objects*). From the DFD point of view, the State Chart is thus hidden behind these models and may not be, in terms of functionality of the system, taken into account.

Figure 36 illustrates some basic consistency relationships between the models described above.

Chapter 7 Digital transformation and process-driven management

As the topics discussed in this textbook show **business process management and information system development are closely tied together**. Moreover, they can be regarded as two opposite sides of the same coin. An information system based on workflow technology is a fatal condition as well as a main development tool of the process-driven organisation. On the other hand, the development of the information system in a process-driven organisation has to be based on the permanent analysis and design of its business processes. One cannot exist without the other. As a final point of this textbook let us look how these main phenomena of the organisation development work together.

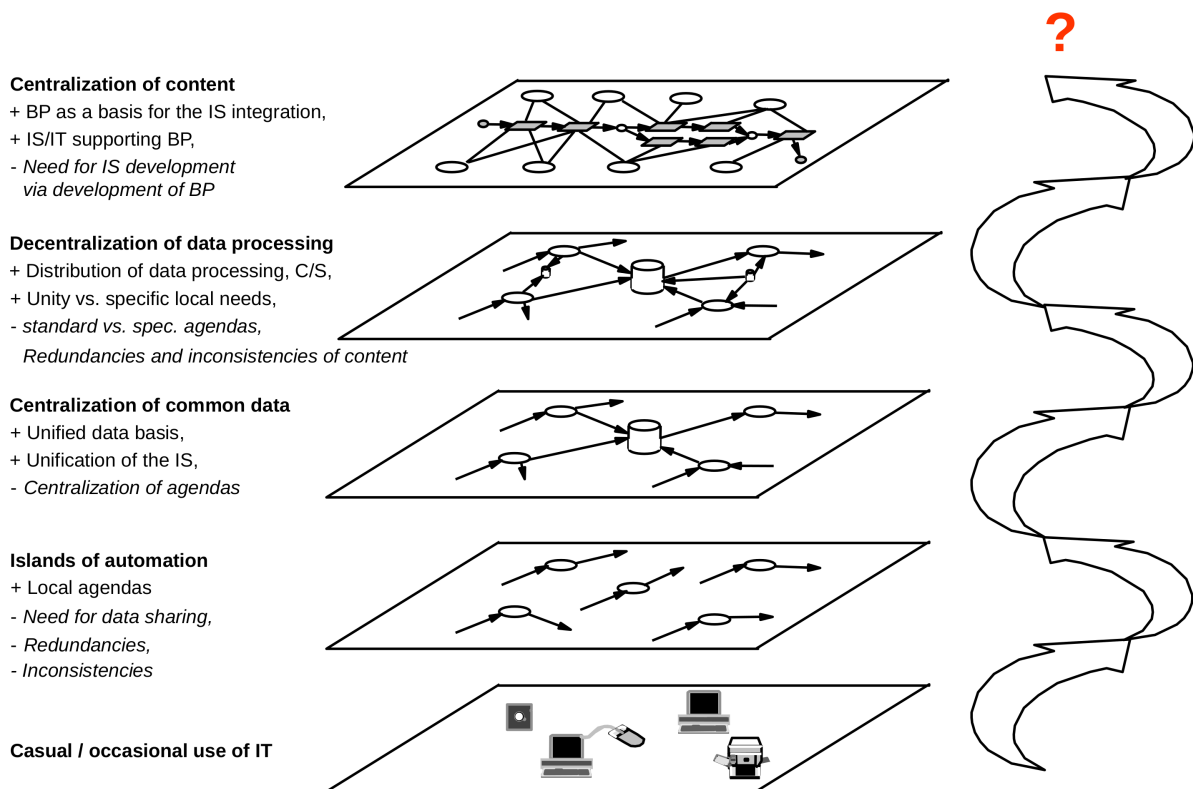


Figure 37: Development of the organisational maturity

Source: Author according to [21]

Figure 37 shows the model of the **evolution of the organisational maturity through its information system**. This idea is based on the work of Richard Nolan ([21]) who firstly expressed the evolution of the information system as a consequence of the growth of the

organisational system via different levels of its maturity. This work later served as a base idea for other “maturity models” like CMM ([4]) for instance.

Nolan discovered the general dependency between the ability to use some kind of technology and the “maturity” of the organisation as a whole, which includes the knowledge and attitudes of its people together with their experience with the previous – lower type of technology (information system – IS). In fact, Nolan's maturity model concept is a creative application of the famous idea of Abraham Maslow ([19]), which defines the hierarchy of “human needs” where fulfilling the needs on a lower level is a prerequisite for the needs of an upper level; a general precondition for the transition to the upper level of the IS is always the perception of the necessity to solve the fatal problems connected with the current level. The model shows how the organisation grows from occasional use of IT through islands of automation to the need for centralisation of data (database system) and consequent need for their partial centralisation, which meets the idea of client-server technology.

Looking at the model at a glance it can be found that the evolution process oscillates between two general aspects: centralisation (integration) and decentralisation (disintegration) in terms of the evolution spiral where the previously surpassed aspects will come back in the future in different, evolutionary higher, forms. According to this principle it can be supposed that the problems with the use of the client-server technology (representing some kind of decentralisation) bring the need for some kind of centralisation. As the figure shows the reason for such a need is the problem of necessary redundancies and consequent inconsistencies of the content of the organisational behaviour following from this technology. The general solution of this problem is the integration (centralisation) of the content via the common definition of business processes which exactly meets the idea of process-driven organisation as well as of connected technology – workflow management systems. It can also be supposed that the future development will undoubtedly bring some problems with this way of centralisation of the content and consequent need for some kind of decentralisation in order to allow necessary different interpretations of the process contents. Current trends to the service-oriented approach to the infrastructural aspects of the organisation can possibly be regarded as a symptom of this need.

As the model shows **business processes represent the important tool for the integration of the information system**. At the same time it also shows how the **information technology serves as a tool for development of the organisation itself** via its information system.

7.1 Process-driven evolution of the organization

In terms of the **process-driven management paradigm** the organisation should be managed in such way that ensures its maximum flexibility in order to accommodate its primary function to the changes in the “turbulent world”. In general, there are two main essential approaches to achieve the above mentioned dynamics of the organisation:

- *Business Process Approach* is characterised with the Business Processes modelling on the one hand, and the Object Life Cycles on the other, thus taking care of their mutual consistency. In this approach, the Object Life Cycles are playing the process-manner description role of “Business Rules” – a process description of crucial restrictions given by business which are naturally static (in spite of the fact that they are described as processes (of object lives)).
 - Advantage of this approach:

Two basic viewpoints of the modelled Real World (the intentional one – business process, versus the static one – object life cycles) allow the dramatic refinement of the set of rules defining the correctness (completeness) of the models.
 - Disadvantage of this approach:

This approach is not open – all possible actions are described in the form of business processes, actors have no chance to function outside of these processes. It means that this approach always reduces the large-scale reality to just the subset defined by the models. This can cause serious restrictions in the ability to change traditional rules, which is still more important in our turbulent world.
- *Legislative Approach* is characterised with the modelling of the objects and their mutual relationships, presuming them to be real world agents with their own activity. We should also take care of the mutual consistency of objects and their life cycles. In this approach, the “Life Cycles” have the role of describing the basic “Real World” rules which have to be respected by any object’s behaviour. Those objects from the model which represent the “Actors” are regarded as “Real World” agents with their own activity. They behave actively and independently, only respecting the rules given to them via their life cycles and mutual dependencies on other objects. Thus, there is no need to model “Business Processes” – the Class Diagram together with models of the life cycles just “delineate the space” for objects’ behaviour – i.e. basic “legislation”. Therefore this approach is called “Legislative”.
 - Advantage of this approach:

This approach is more open and, thus, potentially closer to the reality than the Business Process approach. In the real world, actors usually act according to the rules given by their own activity. All possible methods of action are taken into account.

- o Disadvantage of this approach:

The missing presumption of intentional sets of real world actions, in the form of described business processes, reduces the possibility of formulating integrity rules, which could reduce the set of possible agent actions to only a set of correct ones (in the sense of described business processes). For instance, see the rules for completeness and correctness of object roles for reasons, which are completely useless when the business process description is missing.

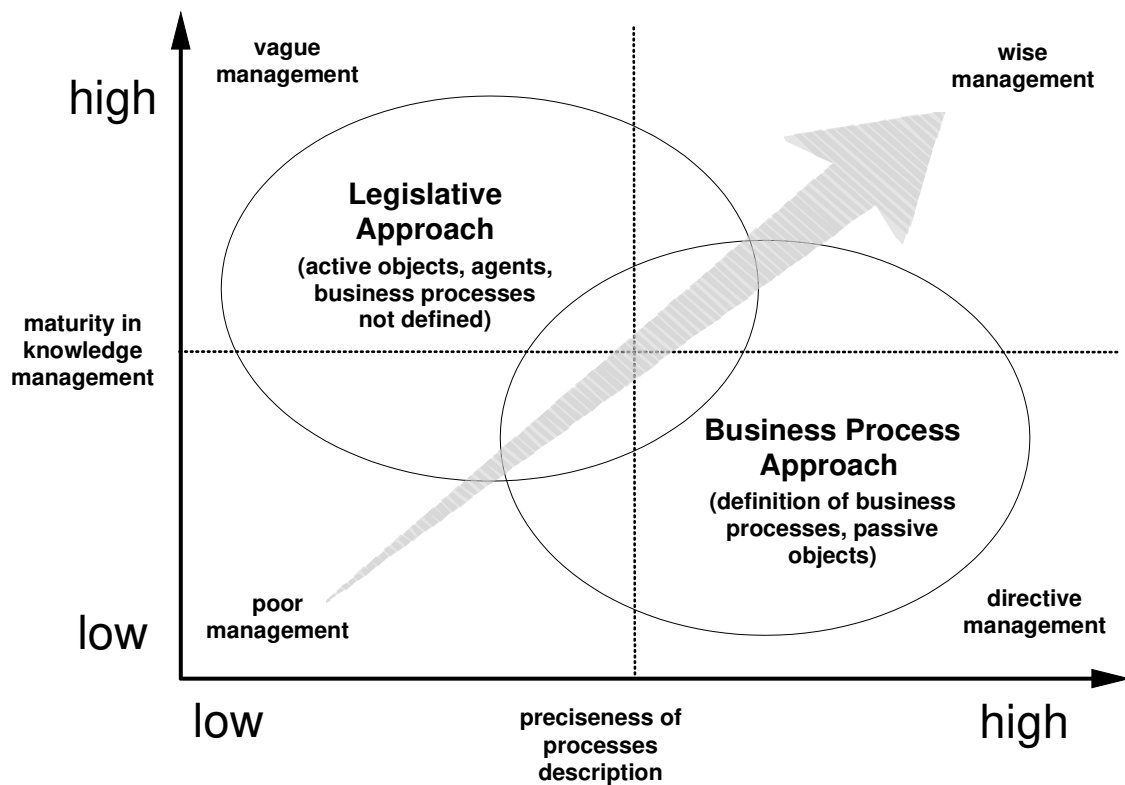


Figure 38: Two main approaches to handle the dynamics of the Real World

Source: Author

Figure 38 shows the context of both the discussed approaches above. The “Legislative Approach” is suitable when the level of maturity in knowledge management is high. As it represents the vague management style, it strongly depends on the self-organising ability of the system. On the other hand, the “Business Process” approach is only suitable when the ability for the real world rules description is high in terms of a relatively stable and well structured environment.

in practice, the correct method lies in the combination of both approaches in order to overcome their limitations and exploit their advantages. In particular, this means the primary ability to find the role of active objects in business processes. This is one of the crucial ideas behind both the further development of the methodology and the way to the organisational maturity of an enterprise.

Bibliography

- [1] Al-Zewairi, M., Biltawi, M., Etaiwi, W., Shaout, A.: Agile software development methodologies: survey of surveys. *J. Comput. Commun.* 05, 74–97 (2017). <https://doi.org/10.4236/jcc.2017.55007>
- [2] Andersson, H., Tuddenham, P.: *Reinventing IT to Support Digitization*, McKinsey & Company (2014).
- [3] Ashworth, C.M.: Structured systems analysis and design method (SSADM). *Inf. Softw. Technol.* 30, 153–163 (1988). [https://doi.org/10.1016/0950-5849\(88\)90062-6](https://doi.org/10.1016/0950-5849(88)90062-6).
- [4] CMM (1995) Carnegie Mellon University Software Engineering institute, Capability Maturity Model: Guidelines for improving the Software Process, 1st ed., Addison-Wesley Professional.
- [5] Coad P., Yourdon E. (1990) “Object-Oriented Analysis”, Prentice-Hall inc., NJ.
- [6] Eriksson, H. E., Penker, M. (2000) “Business modelling with UML: Business Patterns at Work”, Wiley, ISBN 978-0-471-29551-8.
- [7] Hammer M., Champy J. (1994) “Reengineering the Corporation: A Manifesto for Business Evolution”, Harper Business, New York.
- [8] Chen P. P. S. (1976) “The Entity Relationship Model – Towards a Unified View of Data”, *ACM TODS*, Vol. 1 No. 1.
- [9] Jackson, M. A. (1975) “Principles of Program Design”, Academic Press, London.
- [10] Jackson, M. A. (1982) “System Development”, Prentice-Hall inc., Englewood Cliffs, NJ.
- [11] Jackson, M. A. (2002) “JSP in Perspective”; in *Software Pioneers: Contributions to Software Engineering*; Manfred Broy, Ernst Denert eds; Springer.
- [12] Jacobson, i., Booch, G., Rumbaugh, J.: *The Unified Software Development Process*. The Addison-Wesley Object Technology Series. Addison-Wesley, Reading (1999).
- [13] Kobryn, C. (2000) “introduction to UML: Structural modelling and Use Cases”, *Object modelling with OMG – UML Tutorial Series*: www.omg.org.
- [14] Kohnke, O.: it’s not just about technology: the people side of digitization. In: Oswald, G., Kleinemeier, M. (eds.) *Shaping the Digital Enterprise. Trends and Use Cases in Digital innovation and Transformation*, pp. 69–91. Springer international Publishing (2017).
- [15] Lundeberg M., Goldkuhl G., Nilsson A. (1981) “Information Systems Development – A Systematic Approach”, Prentice-Hall inc., Englewood Cliffs, NJ.
- [16] Marca, D., McGowan, C., iDEF (1992), “Business Process and Enterprise modelling, Eclectic Solutions”.
- [17] Marca, D., McGowan, C.: *Structured Analysis and Design Technique*, McGraw-Hill, 1987, ISBN 0-07-040235-3.
- [18] Markovitch, S., Willmott, P.: *Accelerating the Digitization of Business Processes*, pp. 1–4. McKinsey-Corporate Finance Business Practise (2014).

- [19] Maslow, A. (1954) “Motivation and Personality”, New York, Harper & Row, (1954).
- [20] Mayer, R.J., Menzel, C.P., Painter, M.K., deWitte, P.S., Blinn, T., Perakath, B. (1997): iDEF3 Process Description Capture Method Report, Knowledge Based Systems, inc..
- [21] Nolan, R. (1974) “Managing the Four Stages of EDP Growth”, Harvard Business Review.
- [22] OASIS (2006) “Reference Model for Service Oriented Architecture 1.0”, OASIS Standard (on: <http://docs.oasis-open.org/soa-rm/v1.0/>).
- [23] Object Management Group: Unified modelling Language (UML) specification v2.5.1 (2017). Retrieved from <https://www.omg.org/spec/UML/>.
- [24] Object Management Group.: Business Process Model and Notation (BPMN) Specification Version 2.0.2 (2014). <http://www.omg.org/spec/BPMN/>.
- [25] Řepa V. (1995) “Hybrid development methodology”, in Proceedings of 5th Conference on Business information Technology BIT ‘95, Department of Business information Technology, Manchester Metropolitan University.
- [26] Řepa V. (1996) “Object Life Cycle modelling in the Client-Server Applications Development Using Structured Methodology”, Proceedings of the ISD 96 international Conference, Sopot.
- [27] Řepa V. (1999) “information systems development methodology – the BPR challenge”, Proceedings of the ISD99 international Conference, Kluwer Academics, Boise, ID, 1999.
- [28] Řepa V. (2000a) “Process Diagram Technique for Business Processes modelling”, Proceedings of the ISD 2000 international Conference, Kluwer Academics, Kristiansand, Norway, 1999.
- [29] Řepa V. (2003) “Business System modelling Specification”, in: Chu, Hsing-Wei; Ferrer, José; Nguyen, Tran; Yu, Yongquan (eds.). Proceedings of the CCCT2003 international Conference, IIIS, Orlando, FL, 2003.
- [30] Řepa V., Matula M. (2002) “Business Processes modelling with the UML”, Research Paper, Prague University of Economics, Prague, Czech Republic, December 2002.
- [31] Řepa, V. (2000b) “Business System modelling Specification”: <http://opensoul.panrepa.org/meta-model.html>.
- [32] Řepa, V. (2004) “Business Processes and Objects in an Information Systems Development Methodology”. In: Abramovicz, Witold (eds.). Business Information Systems – BIS 2004. Poznan.
- [33] Řepa, V. (2007) “modelling Dynamics in Conceptual Models”. In: Advances in Information Systems Development, ISD 2006 Conference Proceedings, Budapest, New York : Springer.
- [34] Řepa, V. (2008) “Process Dimension of Concepts”. In: Jaakkola, H., Kiyoki, Y., Tokuda, T. Information modelling and Knowledge Bases XiX. Amsterdam : IOS Press.
- [35] Řepa, V. (2011) “Role of the Concept of Service in Business Process Management”. In: Information Systems Development. New York: Springer, LNCS, pp. 623–634, 2011, ISBN 978-4419-9790-6.

- [36] Repa, V., Zeleznik, O. (2014) “Methodological Limitations of modelling Languages BPMN and ARIS” in: Proceedings of the IEEE 15th international Carpathian Control Conference 2014.
- [37] Rosenblueth, A., Wiener, N., Bigelow, J.: Behavior, Purpose and Teleology. In: Philosophy of Science, 10(1943), pp. 18–24.
- [38] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. (1991) “Object-Oriented modelling and Design”, Prentice-Hall inc., Englewood Cliffs, NJ.
- [39] Service Science (2013): “Service Science Management and Engineering”: <http://www.ibm.com/developerworks/spaces/ssme>.
- [40] Scheer, A.–W. (1992) “Architecture of integrated Information Systems – Foundations of Enterprise-modelling”, Berlin.
- [41] Scheer, A.–W. (1994) “Business Process Engineering – Reference Models for industrial Enterprises”, Berlin.
- [42] Svatoš, O.: A4 Method v1.0 (2019). <http://www.e-bpm.org/documents/A4eng.pdf>.
- [43] The Open Group: ArchiMate 3.0 Specification. Van Haren (2016). ISBN 94-01-80047-2.
- [44] The Open Group: TOGAF Version 9.2. Van Haren. (2018). ISBN 978-9401802833.
- [45] Weisman, R. (1999) “introduction to UML Based SW Development Process”: www.softera.com.
- [46] Yourdon, E. (1989) “Modern Structured Analysis”, Prentice-Hall inc., Englewood Cliffs, NJ.

Subject index

- algorithm.....12, 22, 26, 28, 33, 34, 51, 55, 58, 59, 63, 65, 77, 92, 98, 100, 102
- ARIS.....27, 37, 40, 42, 45, 50-53, 55,
- BPMN.....9-12, 30, 31, 39, 40, 45-50, 52, 53, 55, 58
- business event11, 20, 29, 31, 34, 36, 39-43, 45, 48-50, 53, 56-59, 61, 65, 77, 79, 80, 84, 90, 93, 97, 103, 104
- business object.....7, 11, 20, 29, 31, 70, 74, 89, 90, 96
- business ontology.....6, 10-12
- business process management..4, 5, 13, 15, 16, 18, 19, 39, 52-54, 64, 65, 94, 95, 105-107
- causality.....4-8, 10, 11, 65, 69, 70, 93
- conceptual model.....5, 11, 21, 23-27, 29-31, 37, 43, 70-73, 76, 79, 87
- consistency of models.....8-10, 17, 26, 31, 32, 34, 36-38, 53, 60, 61, 78, 86-88, 92, 93, 101-104, 107
- completeness.....5, 9, 10, 37, 38, 78-80, 88, 91, 93, 107, 108
- correctness.....5, 9, 10, 22, 78-80, 93, 107, 108
- structural consistency.....5, 37, 78-80, 86-88
- Data Flow Diagram.....5, 11, 96-104
- digital transformation.....5, 8-11, 94, 105
- Enterprise Architecture.....4, 8, 10, 40, 100
- flexibility.....14, 16, 18, 19, 65, 94, 107
- functionality of IS.....5, 9, 11, 19, 42, 53, 64, 96, 100, 101, 104
- information system.....4-6, 8, 9, 11, 18-23, 27, 36, 43, 44, 53, 77, 94-102, 105, 106
- intentionality.....4, 6-8, 20, 28, 39, 47, 54, 55, 77, 93, 107, 108
- Jackson's theory.....26, 34, 62, 80, 81, 83, 84, 86, 87, 92
- MMABP. 4-8, 10-12, 17, 18, 20, 31, 37, 39, 40, 42, 45, 50, 53-55, 62-64, 69, 70, 74, 101, 104
- Business Modelling Specification.....5, 31, 32

Business Models Consistency package.....	31, 32, 36
Business Process Meta-model.....	29, 31, 34, 35, 42, 43, 52
Business Substance Meta-model.....	31-33, 37
Business Process Flow Pattern.....	55, 56, 59, 65
Principle of Abstraction.....	21, 22
Principle of Different Architectures.....	21, 22
Principle of Modelling.....	20, 21, 27, 99
process abstraction levels.....	5, 39, 40, 62-64
process activity.....	58, 65, 80, 93
Process Map.....	11, 12, 39-41, 65, 66
process state.....	5, 44, 45, 48, 50, 53-55, 58, 61-63, 65, 91, 92
process step.....	55-59, 63, 65-67, 69
negative feed-back.....	54, 55
object life cycle..	5, 11, 25-31, 33, 34, 36-38, 41, 60, 61, 65, 67-70, 74-77, 79, 80, 84-93, 97, 104, 107
organizational maturity.....	105, 106, 109
process state.....	44, 45, 48, 50, 53-55, 58, 61-63, 91, 92
TOGAF Event Diagram.....	11, 31, 39-42
UML.....	9-12, 24, 27, 31-33, 40, 42, 45, 70-74, 84, 92, 98, 100, 101, 104
UML Class diagram.....	11, 12, 25-27, 30, 31, 70-73, 75, 79, 80, 86, 87, 89, 90, 102, 107
UML State Chart.....	11, 12, 30, 31, 70, 74, 75, 80, 84, 85, 90, 92, 104
workflow.....	19, 43, 94-96, 105, 106

Table of Figures

Figure 1: Business system as an equilibrium of intentions and causality Source: Author.....	7
Figure 2: MMABP models Source: Author.....	12
Figure 3: Content, people, and technology as three essential areas of organisation management Source: Author.....	15
Figure 4: Object life cycles versus object model using Jackson's Structure Diagrams, Source: Author.....	26
Figure 5: Two parts of the Real World Business Model Source: Author.....	28
Figure 6: Two views on two dimensions of a business system Source: Author.....	30
Figure 7: Business Modelling Specification overview Source: Author.....	32
Figure 8: Business Substance meta-model package Source: Author.....	33
Figure 9: Business Process Meta-model package Source: Author.....	35
Figure 10: Business Models Consistency package Source: Author.....	36
Figure 11: Global process model (Process Map) in the TOGAF Event Diagram Source: Author.....	41
Figure 12: Types of an event in BPMN Source: [24], http://bpmb.de/poster	49
Figure 13: MMABP Basic Business Process Flow Pattern Source: Author.....	55
Figure 14: Definition of basic blocks and concepts of the Business Process Flow Pattern Source: Author.....	56
Figure 15: Correct business process flow example Source: Author.....	59
Figure 16: Example of a primitive process (Order Receiving) Source: Author.....	62
Figure 17: Example of a complex process Source: Author.....	63
Figure 18: MMABP process abstraction Levels Source <i>MMABP A4 Method</i>	64
Figure 19: Correspondence of Process Map and Process Steps models Source: Author.....	66
Figure 20: Process <i>steps</i> model of the process Client Management Source: Author.....	67
Figure 21: Modified Client Management process Source: Author.....	68

Figure 22: Activity-level sub-process Complaint Handling Source: Author.....	69
Figure 23: Global model of objects (Class Diagram) Source: Author.....	73
Figure 24: Example life cycle of the class Teacher from the conceptual model at Figure 23 Source: Author.....	76
Figure 25: Criteria of completeness and correctness in diagrams Source: Author.....	78
Figure 26: Structure of a file and of a program Source: [11].....	81
Figure 27: Two file structures and a program structure Source: [11].....	82
Figure 28: Structure clash Source: Author.....	83
Figure 29: Structure diagram versus State Chart Source: Author.....	85
Figure 30: Structural coherency of objects and their life cycles Source: Author.....	88
Figure 31: Example of the coherency of models Source: Author.....	91
Figure 32: Relations between process and object models through the object life cycles Source: Author.....	92
Figure 33: information system of a process driven organization Source: Author.....	95
Figure 34: Structure of the Workflow Management System Source: Workflow Management Coalition.....	96
Figure 35: Data Flow Diagram Source: Author according to [46].....	99
Figure 36: Consistency of the model of processes, objects, and DFD Source: Author...	103
Figure 37: Development of the organisational maturity Source: Author according to [21]	105
Figure 38: Two main approaches to handle the dynamics of the Real World Source: Author.....	108

Stránky našeho nakladatelství

<https://oeconomica.vse.cz/>

Název	Information Modelling of Organizations
Autor	prof. Ing. Václav Řepa, CSc.
Vydavatel	Vysoká škola ekonomická v Praze Nakladatelství Oeconomica
Doporučeno	pro magisterské studium na VŠE v Praze
Vydání	první v elektronické podobě
Návrh obálky	Daniel Hamerník, DiS.
Počet stran	117
DTP	Vysoká škola ekonomická v Praze Nakladatelství Oeconomica
Sazba	autor

Zdarma ke stažení

Tato publikace neprošla redakční úpravou.

ISBN 978-80-245-2441-2